

RESEARCH ARTICLE

## A Layered Approach for Merging Application Ontologies: A Case Study from Agriculture Domain

S. A. A. Amarasinghe<sup>1</sup>, Walisadeera Anusha Indika<sup>1\*</sup>, Jeevani Goonetilake<sup>2</sup>

<sup>1</sup>*Department of Computer Science, Faculty of Science, University of Ruhuna, Matara 81000, Sri Lanka*

<sup>2</sup>*Department of Information Systems Engineering, University of Colombo School of Computing, Colombo 00700, Sri Lanka*

**Abstract:** Ontology is a machine interpretable way of representing knowledge in a precise and complete way favorable to solve many problems in the field of knowledge engineering. Different knowledge areas evolve with the time and the applications which use ontologies should be updated with new knowledge accordingly. It is more effective to combine the ontologies with new knowledge with the existing application ontologies rather than designing a new ontology from scratch. When combining ontologies, keeping the original usability of the initial ontologies and the heterogeneity of the components of the ontologies are the main obstacles. As there are no universal standard for naming of ontology components, this is a major reason for the heterogeneity problem. Methods for overcoming these problems are needed. In this research, we have proposed two algorithms to overcome the aforementioned problems. These algorithms for finding correspondences of ontology concepts and merging domain specific application ontologies keeping the original usability of the initial ontologies are the main outcomes of this research. The proposed algorithms are evaluated in terms of accuracy by comparing the resultant ontology merged using the proposed algorithm and the resultant ontology merged by an expert. The evaluation results prove that the proposed methodology merges the domain specific application ontologies very similar to the ontology merged by human intervention.


**Keywords:** *Ontology, Ontology Combining, Ontology Merging, Word2Vec, WordNet*

### Introduction

We have developed an agriculture ontology that represents crop knowledge in order to provide necessary crop information and knowledge to farmers in their context through a mobile-based application and a web-based application (Walisadeera et al., 2015). Now, we need to extend the existing ontology (Crop ontology) with new knowledge areas (e.g. Fertilizer information, Growing Problems, Controlling methods, Harvesting and Post Harvesting information). It is very difficult to modify the existing ontology which is already being used in different applications (mobile-based applications and/or web-based applications) because we need to modify the

queries which are used in different applications based on the modifications of the ontology. Our requirement is to combine the Fertilizer ontology (newly created in a different project) with the existing agriculture ontology (Crop ontology). As the agriculture ontology is used in different applications, one cannot modify their structures when merging the Fertilizer ontology. Another motivation for this research is the need of a course recommender system for students in a university. The purpose of this application is recommending appropriate jobs for the relevant courses that the students follow. The system should link three ontologies such as Job ontology, Student ontology and Course ontology to provide the most

\*corresponding author: [waindika@dcs.ruh.ac.lk](mailto:waindika@dcs.ruh.ac.lk)

 <https://orcid.org/0000-0002-4749-9190>



This article is published under the Creative Commons CC-BY-ND License (<http://creativecommons.org/licenses/by-nd/4.0/>). This license permits commercial and non-commercial reuse, distribution, and reproduction in any medium, provided the original work is not changed in any way and is properly cited.

appropriate job that suits the student considering all the aspects of the courses, student details and the jobs.

By considering above practical scenarios, we had to come up with a practical solution. The ontologies can be merged by changing the structures of them to suit the requirements of the scenarios. This is crucial because we cannot change the structure of every ontology. This is the fact that makes the difference from the usual combining of ontologies. Both of the scenarios we mentioned earlier are already represented in ontologies (fully evaluated and validated ontologies) that use in different applications like mobile-based and web-based. Therefore, changing the structures of them while combining is not an option. There should be a method to merge ontologies without changing their structures, which means, after combining, the ontology should be able to be used in the existing applications too. This means the original usability of the ontologies should be kept in the resulting ontology. The information that could be retrieved using the existing ontology should be retrieved even after combining process as well as can retrieve the additional information based on the merged one as well.

Different techniques for combining ontologies are discussed and implemented in the literature. With the time, the methods used by the researchers to address the problem of ontology combining evolve. In the literature of ontology combining, Klein (2001) mentioned that the different terminologies are used in combining ontologies, such as: *Merging/Integrating*: Constructing a single ontology using two ontologies that have common concepts (overlapping); *Aligning*: Process of determining correspondences between ontology concepts; *Mapping*: Articulating the similarities among ontology concepts belonging to separate ontologies. More simply, it is trying to find the relationships between each pair of concepts of two ontologies. As there are many terminologies for combining ontologies (integrating, aligning, mapping, merging), to be clear, the methodology of this research is used as ontology merging. It will combine two ontologies to make a single final ontology that can be used for many applications.

The main goal of this research is to merge two ontologies into single ontology which keeps the

original usability of both ontologies. As these ontologies are used in different applications by different user communities (e.g., farmers, Agriculture Instructors) who do not have the knowledge about ontologies. When achieving this goal, we must address the following research questions:

- i. *How to check the concept names for correspondences using semantic techniques?*
- ii. *How to overcome the problem of determining the similarity of multi-word phrases written as concept names of ontologies?*
- iii. *What method to use for handling relationships between concepts when merging the parent concepts (super concepts) to the other ontology?*
- iv. *How to keep the original usability of the ontologies in the resulting ontology?*

Ontology merging is a vast research field and it is easy to find the research work that has been done by different researchers to address this problem of merging ontologies. We can consider two ways of combining ontologies. Pinto et al., (1999) mentioned that two kinds of combinations of ontologies as combining ontologies designed for the same domain and combining ontologies from different domains. As we have practical scenarios in the same domain (eg. Agriculture domain), this work will be considered only merging the ontologies in the same domain; for instance, merging ontologies in the domain of agriculture. There is no need of handling instances/individuals while merging ontologies as there are user-friendly tools available for populating ontologies (Akmeemana et al., 2018; Clarkson et al., 2018). Therefore, we do not consider instances when merging ontologies. We only consider the structural changes (only TBox - describes the terminology by relating concepts and roles; not ABox – contains assertions about objects). Since the computational completeness should be maximum when merging ontologies, only OWL-2 DL (Web Ontology Language - 2 Description Logics) ontologies were used for this purpose.

The remainder of the paper is organized as follows. Section 2 presents related research in this field. The

proposed approach is discussed in Section 3. Results and evaluation details are described in Section 4 and 5 respectively. Finally, Section 6 provides a brief summary of the proposed work and some future directions to enhance the approach.

### Related Works

Maree & Belkhatir (2015) have done some work to overcome the semantic heterogeneity problem of different ontologies by merging ontologies. A framework for merging domain specific ontologies exploiting the external knowledge bases is proposed in this work. The main processes of this work are: *Inconsistency checking and resolution in ontologies; Multiple knowledge base assisted ontology merging; Dealing with missing background knowledge and Knowledge base enrichment*. They exploit external knowledge bases like WordNet (<https://wordnet.princeton.edu/>), OpenCyc (<http://www.cyc.com/opencyc/>) and YAGO ([https://en.wikipedia.org/wiki/YAGO\\_\(database\)](https://en.wikipedia.org/wiki/YAGO_(database))) to merge ontologies together into a coherent single ontology. This work compares the relationships between the concepts of ontologies with the relationships between the same concepts that reside in knowledge bases. Different kind of relations like equivalence, disjointness, generalization and specialization are used when corresponding concepts. They have proposed algorithms to merge ontologies. The work has also addressed the scenario of handling missing knowledge. String similarity measures, statistical methods and semantic methods are used when matching the concepts of ontologies. The evaluation of this methodology is performed using different methods and different kinds of ontologies. The methodology has given successful results for every kind of ontology that they have used. However, there are some limitations; for example, it changes the structure of the concept hierarchy when more than one knowledge base has equal relationships for the same pair of concepts. As we use fully evaluated ontologies in our work, this situation is not expected; the string similarity measure that is used for matching concept names lacks intelligence when matching some words.

Heer et al., (2008) addressed the problem of ontology integration in terms of ontology merging in the aim of solving the problems that arise when knowledge

modeling. It is a novel approach for integrating two or more ontologies together. It is an incremental approach which is an interactive one. The user is guided through the process of integration by the developed tool. The tool ensures the integrity of all defined correspondences between the concepts of different ontologies. They focused on the integration in terms of merging. When relating to the concepts of the ontologies to be merged, the types of semantic correspondences are equivalence, overlap, generalization and disjointness. A tool is implemented for assisting the knowledge engineer when choosing the corresponding concepts of the ontologies. The methodology is evaluated for applicability and efficiency using large ontologies used in the building design domain. However, it has some drawbacks. They are: the algorithm can only be used for integrating light-weight ontologies; it assumes that all the ontologies use a common top-level ontology and this method cannot be used by an ontology illiterate person. A knowledge engineer is needed for the process.

Stumme & Maedche, (2001) aimed to overcome the knowledge overlapping problems in common domains in merging domain specific ontologies. In other words, this work also addresses the semantic heterogeneity problem. The method adopted in this research for merging two or more ontologies is using a set of natural language documents and a mathematical approach called Formal Concept Analysis with bottom-up merging. In this process, a lattice of concepts is determined applying the Formal Concept Analysis using the provided natural language documents. The lattice contains the concepts related to the context of the domain of input ontologies. Using two ontology of tourism domain, they have explained how to obtain the concept lattice with the NL (Natural Language) documents, and how to generate the final ontology from the concept lattice. However, it needs a set of domain specific Natural Language documents for the process of concept extraction and the merging process relies on the background knowledge of a domain expert. A knowledge engineer is needed to handle merging conflicts and duplicate concepts merging.

Wang et al., (2018) have proposed a way to align ontologies specific to biomedical domain using entity

definitions and context. They use a neural architecture to encode additional information when available in the process of aligning ontologies. They also use natural language information (textual context) to narrow down an entity’s meaning when the meaning of the entity is not clear. External sources like Wikipedia and scientific articles are exploited when using a supervised learning model to train the ontology alignment model. This work has proved that the derived definitions and contexts can be used effectively when aligning ontologies to obtain good results.

Robin & Uma, (2010) proposed a fully automatic ontology merging approach using hybrid strategy containing both lexical, semantic matching and similarity matching. The semantic matching is done using only WordNet. The WordNet is used to find the meaning of words. WordNet contains every meaning of a considered word. If we consider a word like “bank”, it can be the bank which we deposit money or the bank that is related to rivers. This work only checks synonyms using WordNet and do not evaluate the semantic meaning of a word like “bank”.

By considering above literature, it is clear that there is no such methodology for merging domain specific application ontologies keeping the original usability of the initial ontologies.

### Proposed Approach

We investigated two compulsory tasks to achieve the proposed goal of this research.

These tasks find the answers to the research questions mentioned in the section 1.

The main tasks are:

- i. Matching concepts and finding the correspondences between them
- ii. Merging two ontologies with the correspondences found

We propose a three-layered approach consisting of *input layer*, *concept matching layer* and *ontology merging layer* to achieve the goal of this research. The overview of the proposed methodology is depicted in Figure 1 and functionalities of each layer are explained in sub-sections.

### Finding Correspondences of Ontology Concepts

The ontology designing community has no formal conventions of naming concepts of an ontology when designing ontologies. Ontology designers design ontologies by using different terminologies. Due to the cognitive complexity of matching concepts, most of the time this task is done by a human (Gal & Shvaiko, 2008). As a result, an ontology designer would address this, for instance: “Seedbed” as “Breeding Ground”, leading to the heterogeneity of ontologies of the same domain. The first and the second layers of the methodology (see Figure 1) find the correspondence of the concepts according to the following scenarios.

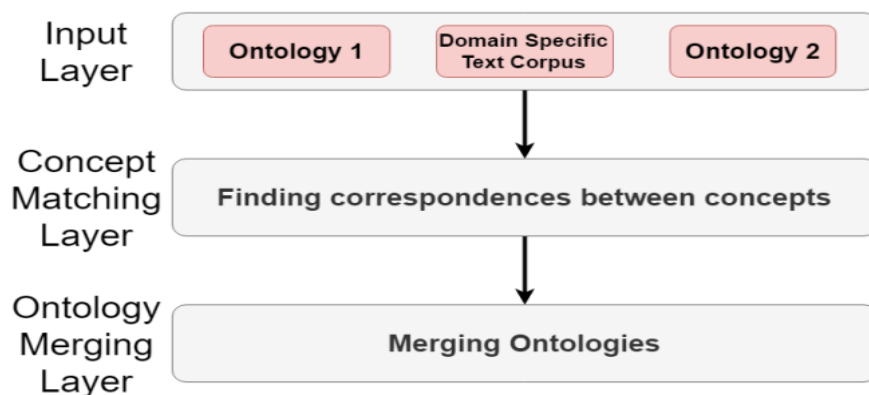


Figure 1: Overview of the Methodology

- *Syntactic and Semantic matching between single-word concept names* (i.e., one word, e.g., Crop, Variety).
- *Syntactic and Semantic matching between multi-word concept names* (i.e., concept names with more than one word, e.g., FarmingStage, AverageYield).

We use the WordNet and the Word2Vec (<https://radimrehurek.com/gensim/models/word2vec.html>) for correspondence of concepts. In the first layer, we prepare all the necessary processes needed for correspondence of the concepts in the second layer. The third layer is responsible for merging two ontologies with the correspondences found in the second layer.

### Input Layer (First Layer)

We input two ontologies with a domain specific text corpus to the layer one. In this layer, we make the ontology with the least number of concepts as the source ontology (base ontology) and the other one as destination ontology. We use this approach to minimize the number of comparisons when checking the correspondence of the concepts of two ontologies, making the process cost-effective manner. A domain specific text corpus is used for determining the semantic relatedness of the multi-word concepts using Word2Vec. It is important for a machine to know the semantic meanings of the words when merging the ontology concepts (Gomaa & Fahmy, 2013). We have used this approach (Word2Vec) for corresponding concepts when the WordNet lacks the concept names and as the solution for heterogeneous multi-word concepts used by different ontology designers, which WordNet does not support. It is difficult to determine the meaning of a phrase as it does not have the simple composition of the meanings of single words (Mikolov, Sutskever, et al., 2013). Using a text corpus to match multi-word concepts is more effective rather than using NLP techniques, which leads to shortage of identifying certain multi-word concept names. The domain specific text corpus is pre-processed and the Word2Vec models are trained for detecting single-word and multi-word concept names. The second layer uses the WordNet and the trained Word2Vec models to correspond the concept names.

### Concept Matching Layer (Second Layer)

When finding correspondence between concepts of the source and destination ontologies, we used the breadth-first traversal for minimizing the number of comparisons. As the concept tree is traversed from top to bottom left to right, if a concept in the top is matched, all the sub concepts of it can be omitted in the next traversal. In this approach, we find whether there is a corresponding concept in the destination ontology for a certain concept in the source ontology. For instance, consider the concepts of the source ontology in the order of breadth-first approach are S1, S2, S2, ..., Sn, and the concepts of the destination ontology in the breadth-first approach are D1, D2, D3, ..., Dn. First, S1 is compared with D1, D2, D3, ..., Dn, then S2 is compared with D1, D2, D3, ..., Dn. This way all the concepts of the source ontology are compared with all the concepts of the destination ontology according to the breadth-first method.

In comparing, the WordNet is used when both concepts of source and destination ontologies are single-word concepts. The meanings of a single word from different semantic aspects are represented in a synset in WordNet. The WordNet uses the similarity of words using cognitive synonyms that is called a synset. The synsets are interlinked by means of conceptual-semantic and lexical relations creating a network of meaningfully related words. We have used the following types of relationships to match the concepts using WordNet.

- **Synonym:** Similar words
- **Hypernym:** A word with broad meaning constituting a category into which words with more specific meanings fall
- **Hyponym:** A word of more specific meaning than a general or superordinate term applicable for it

When these relationships are found in the breadth-first traversal, these concept pairs are stored in a database with the relative relation to be used in the final (third) layer for merging ontologies. The format of the database table is shown below (Table 1).

Table 1: Sample Format of Database Table

Source Ontology Concept	Relationship	Destination Ontology Concept
Crop	Similar	Crop
Harvest	Synonym	Yield
Plant	Hypernym	House Plant
Cow	Hyponym	Herbivore

Source: Author compiled

For the scenarios of one or both concepts of the source and destination ontologies being multi-word concepts, or some concept pairs cannot be corresponded with the relations in WordNet, then we employ the Word2Vec model.

The Word2Vec is a two-layer neural network that processes text. When one inputs a large text corpus into it, it outputs the words represented in a vector space. A word vector represents the meaning of a word in a form of a vector which represents a word as a floating-point number distributed in where semantically similar words are mapped together in the vector space. Word2Vec gives vectors for each word in the input in a way that the semantically similar words have similar vectors. Word2Vec uses the word vectors to perform state of the art syntactic and

semantic similarities for the text provided (Mikolov, Chen, et al., 2013). The built-in skip-gram model of Word2Vec leads the words of a sentence to be surrounded by similar context, making the user who uses Word2Vec can query related words for a given word in the input context. The continuous bag of words (CBOW) model considers a sentence as a bag of words which ignores grammar and the order of precedence of words when constructing a collection of words (Mikolov, Chen, et al., 2013). The skip-gram model predicts the surrounding words given a word and the CBOW model predicts the word based on a given context. The semantic correspondence process of this work is inspired by these two models that are built-in the Word2Vec model, exploiting a rich domain specific text corpus to match the different kind of concept names.

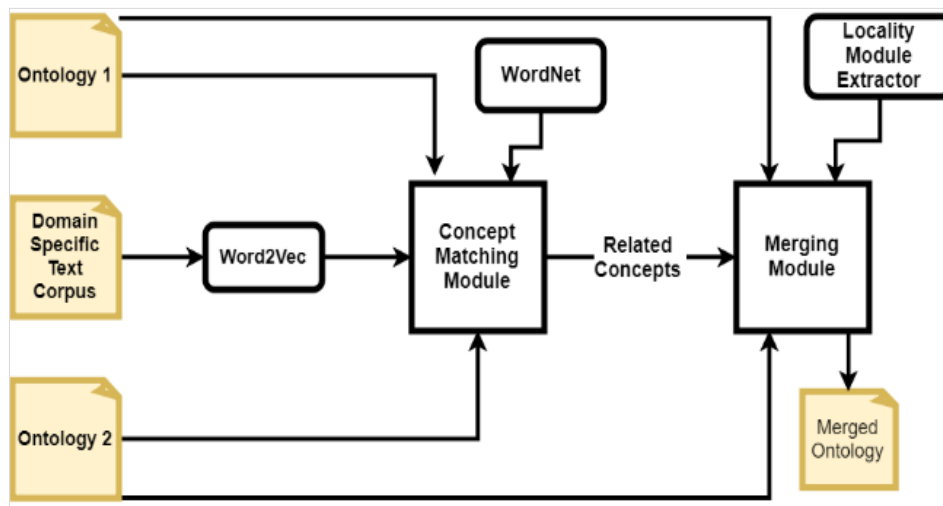


Figure 2: Architecture of the Methodology

Source: Author compiled

In a single comparison which employs Word2Vec, the concept name (this can be single-word or multi-word) of the source ontology is fed into the pre-trained Word2Vec model to get a set of similar words that resides in the same context. The list of words produced by Word2Vec is then compared with the destination ontology concept name using a string similarity measure. The string similarity measure we use in our approach has many advantages over the usual string metrics like Levenstein, Needleman-Wunsch, Jaro-Winkler and Q-Gram that most of the works used in the literature. Fastness, stability and intelligence of this string metric are the reasons for using it in this research (Stoilos et al., 2005). This metric overcomes

the limitations that the usual string metrics have; for instance, Jaro-Winkler metric gives a high score for the two words, “store” and “score”, which are completely two different words. The string metric proposed by Stoilos et al., (2005) is mainly used to overcome this limitation.

The matched concept pairs using the Word2Vec model is stored in the database using a different relationship as same as the format shown in Table 1. In this way, all the concepts of two ontologies are compared and then the identified correspondences are stored in a database. Figure 2 shows the overall architecture of the methodology.

**Input:** Two Ontologies, domain specific text corpus

**Output:** Pairs of corresponded concepts with relations

1. Pre-process the text corpus and train the word2vec model using the preprocessed text corpus.
2. Make the ontology with the lesser number of concepts as the source ontology, call it O1. If the number of concepts is equal, choose arbitrarily.
3. Make the other ontology as the destination ontology, call it O2.
4. Traverse O1 in breadth first way, call the current concept C1.
5. Traverse O2 in breadth first way, comparing C1 with each concept of C2.
6. When comparing (ignore the concepts that are already compared),
  - If both concept names are single worded,
    - If C1 is similar to C2, store in the database with similar relationship.
    - Else if C1 is a synonym of C2, store in the database with synonym relationship.
    - Else if C1 is a hypernym of C2, store in the database with hypernym relation.
    - Else if C1 is a hyponym of C2, store in the database with hyponym relation.
    - Else input C1 and C2 into pre-trained word2vec model for most similar words.
      - Compare C2 with each word of the results of word2vec using a string metric.
        - If match, store in the database with equal relationship.
        - else ignore C1.
  - else one or both of concept names are multi worded,
    - Input C1 and C2 into pre-trained word2vec model for similar phrases.
      - If match, store in the database with equal relationship.
7. Repeat from step 4 until all the concepts of the source ontology is compared.

Figure 3: Algorithm for finding correspondences.

In third layer (Ontology Merging Layer), the two ontologies are merged using the identified correspondences. The proposed algorithm for finding correspondences between the ontology concepts is shown in Figure 3.

### Ontology Merging Layer (Third Layer)

The main goal of merging ontologies in this work is not to change the initial structures of the ontologies after merging. After merging, the ontology should be able to be used in the existing applications. Further, information that could be retrieved using the existing ontology should be retrieved even after the merging process as well as can retrieve the additional information based on the merged one. Therefore, we do not change the initial structures when merging them. By satisfying this requirement, we addressed three structural merging scenarios as follows:

- Child (sub class) to Child (sub class) Merging
- Parent (super class) to Parent (super class) Merging
- Parent (super class) to Child (sub class)/Child (sub class) to Parent (super class) Merging

According to the relationship of the correspondence between the concept pair, the merging is done. For the above mentioned three scenarios, there can be any relationship type such as similarity, synonym, hypernym, hyponym or equal. Hence, there can be altogether 15 scenarios of merging. Here, we map the relationships to the method of merging as follows, by considering the source ontology concept as C and the destination ontology concept as D:

- **Similarity** (i.e., both ontologies have same concepts): *transfer all axioms, constraints, restrictions of C to D.*
- **Synonym or Equal** (i.e., concept in source ontology is a synonym of the concept in destination ontology): *make C equivalent to D.*
- **Hypernym** (i.e., concept in source ontology is a superordinate of the concept in destination ontology): *make C a superclass of D.*
- **Hyponym** (i.e., concept in source ontology is a subordinate of the concept in destination ontology): *make C a subclass of D.*

When merging ontologies in different scenarios, we might need to merge a whole sub-tree of the source ontology to a concept in the destination ontology. With the sub-tree, the whole set of axioms of every sub class of the sub-tree are needed to be transferred to the concept of the destination ontology. Sometimes there may be relations that go from the sub-tree to different concepts that are not in the sub-tree. We use the locality module extractor to find the relationships around the concept and alter them and set them as axioms of the other ontology. The locality module extractor extracts all the domains and ranges of the relations that goes out of the sub-tree and comes into the sub-tree along with all the subclass-superclass relationships. For example, consider the concept “Area” of the source ontology and has sub-concepts as Costal, Hill and Zone. First, it gets all axioms (subClassOf axioms) related to concept “Area”. Then all the axioms with the concept “Area” are altered and placed to the concept “Area” of destination ontology. Now, all the altered axioms are added to the destination ontology. The concept “Area” of source ontology is merged to the concept “Area” of destination ontology. The extractor is built on top of the OWL API which we use to manipulate the ontology constructs when merging to extract the logic-based modules from a given ontology (Jiménez-Ruiz et al., 2008). This makes the merging process efficient in a programmatic perspective as it automatically extracts all the relevant axioms of a given sub-tree of the ontology. The main reason for employing this in our research is the need of merging sub-trees to the destination ontology. When merging ontologies as mentioned, it is assured that the structures are not changed. After the merging process, a reasoner attached to the ontology is used to check whether there is any inconsistency. The resulting ontology can contain concepts with multiple inheritance (concepts with more than one parents). This makes ontologies harder to maintain manually. However, the multiple inheritances make the ontologies richer with better axiomatizations (Mikel Egaña Aranguren, 2010). The user can get the final merged ontology as an owl file to be used with any application they used previously. The ontology merging algorithm is depicted in Figure 4. According to this algorithm, two scenarios such as *Child to Child and Parent to Parent* with respect to the Similarity Relationship, are depicted in Annex 01 and



Annex 02 respectively. We identify a child concept if it has no sub-concepts and a parent class if it has one or more sub-concepts. Since we are doing breadth first traversing, during the loop, only the valid if-conditions will be executed making the comparisons minimal. According to the proposed algorithm, it will first choose first concept of the source ontology as C1 and the first concept of the destination ontology as C2.

Then it checks whether which of following conditions are valid for C1 and C2:

- C1 and C2 both are child concepts.
- C1 is a child and C2 is a parent.
- C1 is a parent and C2 is a child.
- C1 and C2 both are parents.

*Algorithm for Merging Ontologies with the Found Correspondences*

**Input:** Two ontologies and matched concept pairs with relations

**Output:** Merged ontology

1. Take the first pair of concepts, call the concept from ontology1 C1 and ontology2 C2.
2. Check whether the C1 and C2 is a child or parents.
  - If C1 is a child and C2 is a child,
    - If relationship is similarity  
transfer all axioms, restrictions and constraints of C1 to C2
    - If relationship is synonym or equal  
make C1 and C2 equivalent classes.
    - If relationship is hypernym  
make C1 a superclass of C2
    - If relationship is hyponym  
make C1 a subclass of C2
  - If C1 is a child and C2 is a parent,
    - If relationship is similarity  
transfer all axioms, restrictions and constraints of C1 to C2
    - If relationship is synonym or equal  
make C1 and C2 equivalent classes
    - If relationship is hypernym  
make C1 a superclass of C2
    - If relationship is hyponym  
make C1 a subclass of C2
  - If C1 is a parent and C2 is a parent
    - If relationship is similarity  
transfer all axioms, restrictions and constraints of C1 to C2
    - If relationship is synonym or equal  
make C1 and C2 equivalent classes
    - If relationship is hypernym  
make C1 a superclass of C2
    - If relationship is hyponym  
make C1 a subclass of C2
  - If C1 is a parent and C2 is a child
    - If relationship is similarity  
transfer all axioms, restrictions and constraints of C1 to C2
    - If relationship is synonym or equal  
make C1 and C2 equivalent classes
    - If relationship is hypernym  
make C1 a superclass of C2
    - If relationship is hyponym  
make C1 a subclass of C2
3. Take next matched concept pair and repeat step 2 until all concept pairs are merged.
4. Output the merged ontology as an owl file.

Figure 4: Ontology Merging Algorithm

Let's consider C1 and C2 are parent concepts, and the scenario is for similarity relationship. The algorithm checks whether the correspondence between two concepts is similarity, synonym (or equal), hypernym or hyponym. If the correspondences in the database have no correspondence like this (correspondences stored in the database in the first layer of the implementation, the correspondences obtained using the trained Word2vec and from WordNet), then the merging of these two concepts will be skipped. Then, in the second loop, according to the breadth first traversal, the same process will be repeated. Then if the training happened accurately in the previous layer, there should be a correspondence in the database which says "C1" is equal to "C2". So, when the algorithm detects this, it will transfer all axioms, restrictions and constraints of C1 to C2, merging two concepts.

Consider a scenario which uses hypernym relations. At some point in the loop, C1 can become "ControlMethod" of the source ontology and C2 can become "WeedControl" in the destination ontology. We clearly see that "WeedControl" can be a subconcept of "ControlMethod". If the training was accurate in the previous step, a correspondence should be there in the database that says that "ControlMethod" is a hypernym (superconcept) of "WeedControl". If it exists in the database, the algorithm proceeds with making C2 is a subconcept of C1.

## Results

In related works, most of the findings with respect to the correspondences of the concepts of ontologies are done by humans. This makes the work of merging ontologies more labor intensive. According to the goal of this work, user friendliness is one of the factors we address because of the ontology-based applications are not always used by the people who have knowledge of concepts of ontologies. They may be agricultural instructors, experts in agriculture sector and many more professions in other sectors. The proposed methodology is an automatic approach which can be used by any user without the knowledge of ontologies. The only requirement is a rich domain specific text corpus in addition to two ontologies to be merged. The

algorithm we proposed for finding correspondences in Figure 3 that answers first and second research questions mentioned in section 1. The external source we employed to find correspondences (WordNet) and the domain specific text corpus we used along with Word2Vec model helps to match the concepts using both lexical and semantic aspects. The Word2Vec also helps to find the semantic similarities of the word phrases.

Another outcome of this research is the ontology merging algorithm in Figure 4 which satisfies our goal of keeping the original usability of the ontologies even after the merging is done. This algorithm overcomes the third and fourth research questions mentioned in section 1.

## Validation and Evaluation

We have utilized the agricultural ontologies in the domain of agriculture for the evaluation. The original usage of the ontologies is kept intact with the suggested merging process by following the breadth first traversal. In the merging process, none of the original concept names are changed, therefore, keeping the original usage. For evaluating whether we kept the original usage in the final merged ontology, we can run the same ontology queries in both input ontologies and in the final merged ontology and see they work as expected.

We evaluated the proposed methodology by comparing the output of the methodology with the results of expert merging. We used two agricultural ontologies to evaluate the merging process. The source ontology and destination ontology consist of 18 concepts separately. Figures 5 and 6 show these ontologies. Merging with the help of the expert, the following results were obtained.

Number of correspondences: 9

Number of Similar concept pairs: 4

Number of Synonym concept pairs: 4

Number of Hypernym relations: 1

Number of Hyponym relations: 0

The expert was not allowed to change the structures of the ontologies or the concept names. The expert was given the same rules as the algorithm but not the traversal method. The expert was given the freedom to merge the ontologies with every possible correspondence. The proposed algorithm gave the following results after matching the concepts.

- Number of correspondences: 6
- Number of Similar concept pairs: 4
- Number of Synonym concept pairs: 1
- Number of Hypernym relations: 1
- Number of Hyponym relations: 0

We received the percentage of correspondences, the algorithm successfully identified merged against the correspondences identified and successfully merged by the expert. This gave an accuracy percentage of 66.66% which is a promising percentage.

As we have done this research to improve the ontology proposed by Walisadeera et al., (2015) with the ontologies that may have designed for the sub domains of agriculture for obtaining new knowledge we only used the ontologies related to the agriculture domain. The accuracy of this merging process completely depends on the correspondences we obtain from training of the corpus (i.e. it depends on WordNet and Word2Vec). Since WordNet is a knowledge base, it has its own limitations. But, since Word2Vec is an algorithm, we can feed as many texts we want to train the model. Word2Vec gives more accurate results when the text we feed into it gets bigger. For example, if we can see that “ControlMethod” and “WeedControl” can be made superconcept and subconcept, if this correspondence is not identified by WordNet or Word2Vec, those two concepts will not be merged by our algorithm. So, the proposed algorithm completely depends on WordNet and Word2Vec results. The methodology is also applicable for merging ontologies of other domains as well with a rich domain specific text corpus.

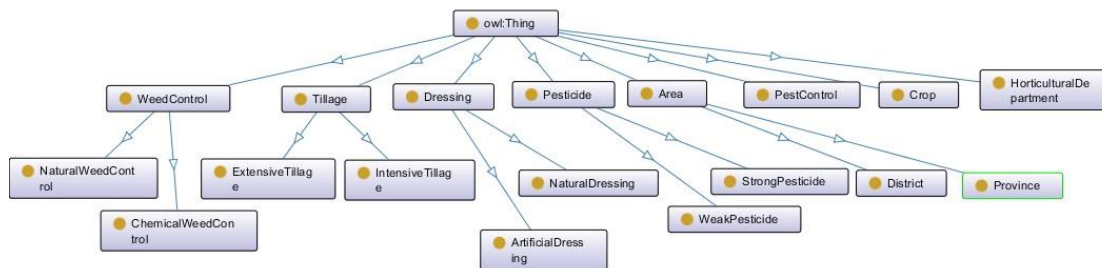


Figure 5: Source Ontology

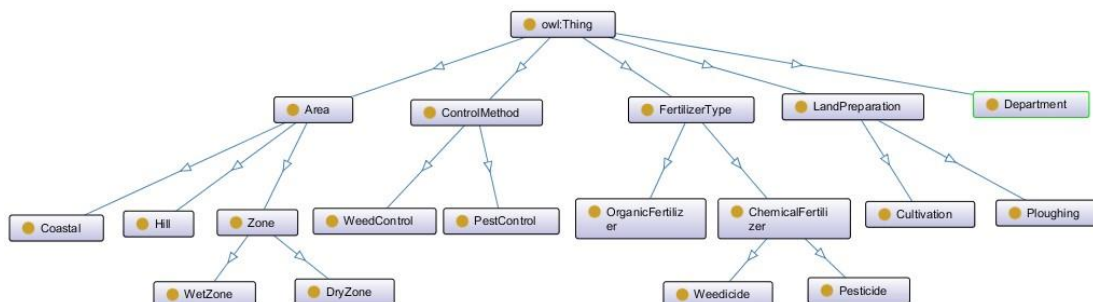


Figure 6: Destination Ontology

## Conclusion

This research is motivated by practical problems in different application domains. One scenario is the problem of scaling up the existing agricultural ontology proposed by Walisadeera et al., 2015 for farmers in Sri Lanka to merge with new knowledge. Then they can get more information in the form of the existing ontologies which linked with its applications like “Govi-Nena” mobile-based application that is linked with the crop ontology. As a solution, in this research, new algorithms were introduced to merge two ontologies. One of the proposed algorithms presented in Figure 3 is for finding concept correspondences. Another outcome of this research is the ontology merging algorithm in Figure 4 which satisfies our main goal of keeping the original usability of the ontologies even after the merging is done. In conclusion, this work addressed the goal of the research successfully in merging application ontologies by keeping the original structures of the ontologies intact.

With the future improvements, the methodology will be used in more advanced and complex ontologies. As a future work, the correspondence finding process can be improved using an extra layer of intelligence to determine the abbreviations and half word concept names. Moreover, an information extraction method can be used to address the concepts which are not correspondence. The concepts that can be related will be able to merge introducing new relations.

## References

- Akmeemana, R. A. O. M. P. D., Walisadeera, A. I., Goonathilake, M. D. J. S., & Ginige, A. (2018). A Semi-automatic Approach to Collaboratively Populate an Ontology for Ontology-Illiterate Users. *In International Conference on Computational Science and Its Applications*, 120–135.
- Clarkson, K., Gentile, A. L., Ghul, D., Ristoski, P., Terdiman, J., & Welch, S. (2018). User-Centric Ontology Population. *In European Semantic Web Conference*, 112–127.
- Gal, A., & Shvaiko, P. (2008). Advances in ontology matching. *In Advances in Web Semantics I*, 176–198.
- Gomaa, W. H., & Fahmy, A. A. (2013). A survey of text similarity approaches. *International Journal of Computer Applications*, 68(13), 13–18.
- Heer, T., Retkowitz, D., & Kraft, B. (2008). Tool support for the integration of light-weight ontologies. *In International Conference on Enterprise Information Systems*, 175–187.
- Jiménez-Ruiz, E., Grau, B. C., Sattler, U., Schneider, T., & Berlanga, R. (2008). Safe and economic re-use of ontologies: A logic-based methodology and tool support. *In European Semantic Web Conference*, 185–199.
- Klein, M. (2001). Combining and relating ontologies: an analysis of problems and solutions. *In IJCAI-2001 Workshop on Ontologies and Information Sharing*, 53–62.
- Maree, M., & Belkhatir, M. (2015). Addressing semantic heterogeneity through multiple knowledge base assisted merging of domain-specific ontologies. *In Knowledge-Based Systems (73rd ed., pp. 199–211)*.
- Mikel Egaña Aranguren. (2010). *Automatic maintenance of multiple inheritance ontologies*. <http://ontogenesis.knowledgblog.org/49>
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*. arXiv:1301.3781. <https://arxiv.org/pdf/1301.3781.pdf>
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *In Advances in Neural Information Processing Systems*, 3111–3119.
- Pinto, H.S., Go´mez-Pe´rez, A. and Martins, J.P. (1999) Some issues on ontology integration. *In Proceedings of the Workshop on Ontologies and Problem Solving Methods IJCAI-99*, Stockholm, Sweden. <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.46.6489&rep=rep1&type=pdf>
- Robin, C. R., & Uma, G. V. (2010). A novel algorithm for fully automated ontology merging using hybrid strategy. *European Journal of Scientific Research*, 47(1), 74–81.
- Stoilos, G., Stamou, G., & Kollias, S. (2005). A string metric for ontology alignment. *In*

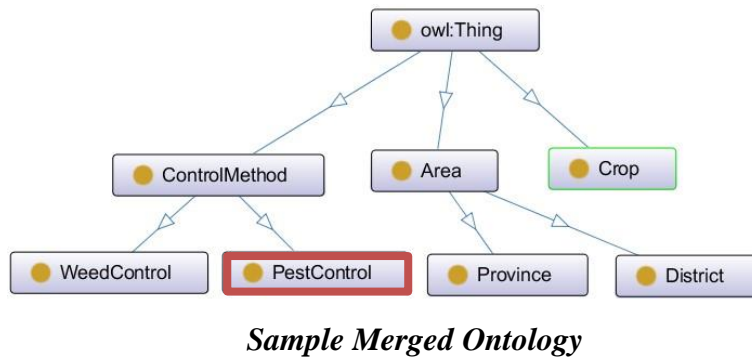
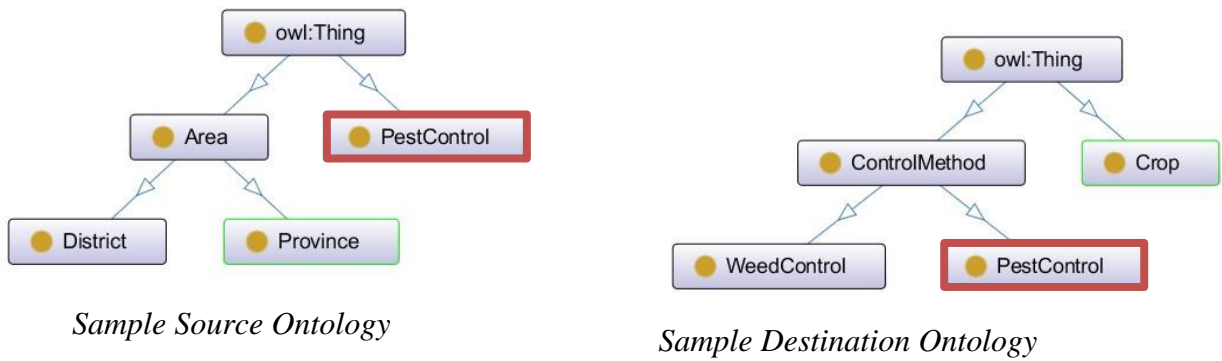
*International Semantic Web Conference*, 624–637.

Stumme, G., & Maedche, A. (2001). FCA-Merge: Bottom-up merging of ontologies. *In IJCAI, 1*, 225–230.

Walisadeera, A. I., Ginige, A., & Wikramanayake, G. N. (2015). User centered ontology for Sri Lankan farmers. *Ecological Informatics, 26*, 140–150.

Wang, L.; Bhagavatula, C.; Neumann, M.; Lo, K.; Wilhelm, C.; and Ammar, W. 2018. Ontology alignment in the biomedical domain using entity definitions and context. In *Proceedings of the BioNLP 2018 workshop*, 47–55.

**Annex 01: Similarity Relationship: Child to Child Scenario**



**Annex 02: Similarity Relationship: Parent to Parent Scenario**

