Title : Dynamic ontology for service robots

Name : Saranya Kanjaruek

Dynamic Ontology for Service Robots

Saranya Kanjaruek

Ph.D.

2017

UNIVERSITY OF BEDFORDSHIRE

Dynamic Ontology for Service Robots

by

Saranya Kanjaruek

A thesis submitted to the University of Bedfordshire in partial fulfilment of the requirements for the degree of Doctor of Philosophy

February 2017

DYNAMIC ONTOLOGY FOR SERVICE ROBOTS

SARANYA KANJARUEK

ABSTRACT

Automatic ontology creation, aiming to develop ontology without or with minimal human intervention, is needed for robots that work in dynamic environments. This is particularly required for service (or domestic) robots that work in unstructured and dynamic domestic environments, as robots and their human users share the same space. Most current works adopt learning to build the ontology in terms of defining concepts and relations of concepts, from various data and information resources. Given the partial or incomplete information often observed by robots in domestic environments, identifying useful data and information and extracting concepts and relations is challenging. In addition, more types of relations which do not appear in current approaches for service robots such as “HasA” and “MadeOf”, as well as semantic knowledge, are needed for domestic robots to cope with uncertainties during human–robot interaction. This research has developed a framework, called Data-Information Retrieval based Automated Ontology Framework (DIRAOF), that is able to identify the useful data and information, to define concepts according to the data and information collected, to define the “is-a” relation, “HasA” relation and “MadeOf” relation, which are not seen in other works, to evaluate the concepts and relations. The framework is also able to develop semantic knowledge in terms of location and time for robots, and a recency and frequency based algorithm that uses the semantic knowledge to locate objects in domestic environments. Experimental results show that the robots are able to create ontology components with correctness of 86.5% from 200 random object names and to associate semantic knowledge of physical objects by presenting tracking instances. The DIRAOF framework is able to build up an ontology for domestic robots without human intervention.

DECLARATION

I declare that this thesis is my own unaided work. It is being submitted for the degree of Doctor of Philosophy of University of Bedfordshire.

It has not been submitted before for any degree or examination in any other university.

Name of candidate: MS. SARANYA KANJARUEK

Signature: Saranya Kanjaruek.

Date: February, 7th 2017

Acknowledgements

I would like to thank my supervisor Prof. Dayou Li for his encouragement, suggestions and support. Since I started the Ph.D., he has taught me how to become a good researcher and lecturer with kindness. He helped me overcome lots of difficult problems. I am lucky to have him as the best supervisor in my professional life. I also would like to thank my viva examiners, Dr. Marc Conrad and Dr. Jianhua Shao for their very helpful comments and suggestions. I also thank Khon Kaen University and the campus in Nongkhai for supporting the scholarship for this study and the Institute for Research in Applicable Computing at the University of Bedfordshire for offering me a place to study.

I would like to thank my family for supporting, guiding and teaching me from the beginning and encouraging until I achieve the highest degree of study. I would like to say thank you for the support and cheerfulness of my husband, Dr. Noppakun Boonsim. Finally, I would like to thank Dr. Peter Norrington for support with proofreading the thesis.

LIST OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ALC	Attributive Concept Language with Complements
ATCT	Automatic Taxonomy Construction from Text framework
CC	Conjunction
CN	Cardinal number
Comp	Computed retrieval
csc	Common semantic cotopy
DIRAOF	Data-Information Retrieval based Automated Ontology Framework
DR	Data retrieval
DT	Determiner
IFR	International Federation of Robotic
IN	Preposition
IR	Information retrieval
IRI	Internationalized Resource Identifiers
KB	Knowledge base
LFU	Least frequently used
LRU	Least recently used
NLP	Natural Language Processing
NNP	Singular proper nouns
NNPS	Plural and proper nouns
O_C	Core Ontology
OMCS	Open Mind Common Sense
O_R	Reference Ontology
ORO	Open Robot Ontology
OWL	Web Ontology Language
OWL 2	OWL 2 Web Ontology Language
RB	Adverb
RDF	Resource Description Framework

Ref	Reference retrieval
RelExt	A relation from an existing ontology
RT	Robot Technology
sc	Semantic cotopy
SEP	*Stanford Encyclopedia of Philosophy*
SP	Semantic Precision
SR	Semantic Recall
TFS	Types Feature Structure
TP	Taxonomy Precision
TR	Taxonomy Recall
UH	Interjection
URI	Uniform Resource Identifiers
URL	Uniform Resource Locator
VB	Verb
XML	eXtensible Markup Language
XSD	XML Schema Definition

LIST OF NOTATIONS

Chapter 2

$\Im$	Intension of concept
Σ	A set of concept instance
L	Linguistic realization
W	Weighting vector
W_b	Frequency from books
W_c	Reliability score from ConceptNet
W_g	Frequency from Google 1-T5 gram
W_n	Similarity score of two objects based on WordNet
W_u	User evaluation score
(n_1, n_2)	Connection between two concepts n_1 and n_2
$W_b = f\,(n_1, n_2)$	Frequency of the connection (n_1, n_2) is extracted from books.
$W_c = \text{score}\ (n_1, n_2)$	The reliability score of the assertion between n_1 and n_2 from ConceptNet
$W_g = f_{gram}\ (n_1, n_2)$	The frequency of the n-grams that contain n_1 and n_2 from the Google Web 1T 5-gram
$W_n = d_{WordNet}\ (n_1, n_2)$	The distant between two nouns n_1 and n_2 in WordNet

$P(x\|y)$	Probability of concept *x* on concept *y*
t	A co-occurrence threshold value
$score(p, x)$	Score of parent *p* of concept *x*
$P(a\|x)$	Probability of ancestor *a* on concept *x*
A_p	A list of ancestors *p*
$w(a, x) = \frac{1}{d(a, x)}$	Weight value of node x and ancestor a
d (a, x)	The distance between node x and ancestor node a
T_C	Core taxonomy
T_R	Reference taxonomy
C_C	The concept of core taxonomy
C_R	The concept of reference taxonomy
$SP(T_C, T_R) = \frac{\|C_C \cap C_R\|}{\|C_C\|}$	Semantic precision
$\mathrm{csc}(c, T_C, T_R) = \{c_i \| c_i \in C_C \cap C_R \wedge (c_i \leq C_C\ c \vee c \leq C_C\ c_i)\}$	Common semantic cotopy of a concept and its super-concept and sub-concept
$tp_{csc}(c, T_C, T_R) = \frac{\|\mathrm{csc}(\mathrm{c}, \mathrm{T}_\mathrm{C}, T_R) \cap \mathrm{csc}(c, T_R, T_C)\|}{\|\mathrm{csc}(c, T_C, T_R)\|}$	Taxonomic precision
$tr_{csc}(c, T_C, T_R) = \frac{\|\mathrm{csc}(\mathrm{c}, \mathrm{T}_\mathrm{C}, T_R) \cap \mathrm{csc}(c, T_R, T_C)\|}{\|\mathrm{csc}(c, T_R, T_C)\|}$	Taxonomic recall
$TF(T_C, T_R) = \frac{2 \cdot TP_{csc}(T_C, T_R) \cdot TR_{csc}(T_C, T_R)}{TP_{csc}(T_C, T_R) + T_{csc}(T_C, T_R)}$	Taxonomic F-measure

$P_{n_l^w}(n_{l+1}^w\|o)$	A probability of node n_{l+1}^w on object *o*
$\psi(w,\mathcal{H},p) = \sum_{l=1}^{L}\frac{\emptyset(v,v_l)}{L}$	Decision function to match score between a path *p* and word *w* given the hierarchy *H*
$p^w = \arg\max_{\mathrm{p}} \psi(w,\mathcal{H},p)$	A correct path in hierarchy $\mathcal{H}$
$\emptyset(v,v_l) = -(v-v_l)^T(v-v_l).$	Matching score between word *v* and v_l
$P(Ref,Comp) = \frac{\|Comp\cap Ref\|}{\|Comp\|}$	Precision of retrieval (*Ref*) with a computed (*Comp*) ontology
$R(Ref,Comp) = \frac{\|Comp \cap Ref\|}{\|Ref\|}$	Recall of retrieval (*Ref*) with a computed (*Comp*) ontology
$F_1(Ref,Comp) = \frac{2\cdot P(Ref,Comp)\cdot R(Ref,Comp)}{P(Ref,Comp)+R(Ref,Comp)}$	F-measure of retrieval (*Ref*) with a computed (*Comp*) ontology
$LP(O_C,O_R) = \frac{\| C_C \cap C_R\|}{\|C_C\|}$	Lexical precision of core ontology (C_C) and reference ontology (O_R)
$LR(O_C,O_R) = \frac{\| C_C \cap C_R\|}{\|C_C\|}$	Lexical recall of core ontology (C_C) and reference ontology (O_R)
$tp_{ce}(c_1,c_2,O_C,O_R) := \frac{\|ce(c_1,O_C)\cap ce(c_2,O_R)\|}{\|ce(c_1,O_C)\|}$	Taxonomy precision of core ontology and reference ontology
c_1, c_2	A concept of core ontology and a concept of reference ontology
$sc(c,O) := \{c_i\|c_i \in C \wedge (c_i \leq c \vee c \leq c_i)\}$	Semantic cotopy calculation of all super-concepts and sub-concepts

$csc(c,O_1,O_2) := \{c_i \mid c_i \in C_1 \cap C_2 \wedge (c_i < 1\ c \vee c\ < 1\ c_i)\}$	Common semantic cotopy calculation of concept and concept hierarchy
$TP(O_C,O_R) := \frac{1}{\lvert C_C \rvert}\sum_{c\in C_C}\left\{\begin{array}{l} tp(c,c,O_C,O_R) \\ max_{c'\notin C_R} tp(c,c',O_C,O_R)\end{array}\right.$	Taxonomy precision of core ontology (C_C) and reference ontology (O_R)
$TP_{csc}(O_C,O_R) := \frac{1}{\lvert C_C \cap C_R \rvert}\sum_{c\in C_C \cap C_R} tp_{csc}(c,c,O_C,O_R)$	Taxonomy precision of common semantic cotopy measurement
$TR_{csc}(O_C,O_R) := TP_{csc}(O_R,O_C)$	Taxonomy recall of common semantic cotopy measurement
$volatility(z) = \frac{1}{count(P,Q)}\sum_{\forall P,Q} v'(P,Q)$	Volatility score measurement of instance *P* and *Q*
$violation_g(O) = \frac{1}{count(S,G)}\sum_{\forall S,G} gv(S,G)$	Violation score measurement of instance *S* and parent *G*

Chapter 5

$F_{\text{frequency}}(x) = 1$	The weight function of frequency
$F_{\text{recency}}(x) = 0.5^{(\text{NFileN}-x)}$	The weight function of recency
$F(x)$	The weight function
$D_{loc} = \frac{F_{loc}}{NFileN} + (\ (0.5)^{NFileN-RFileN} \times \frac{RFileN}{NFileN})$	Decision score of location

Chapter 6

$Sim(C_1,C_2) = \frac{2\ \times depth(LCS(c_1 c_2))}{x_1 +\ x_2\ + 2\ \times depth(LCS(c_1,c_2))}$	The semantic similarity score between concept 1 (c_1) and concept 2 (c_2)

CHAPTER 1 INTRODUCTION

1.1 Background and Motivations

According to the International Federation of Robotics (IFR), a service robot operates semi or fully autonomously to perform services useful to the well-being of humans and equipment, excluding manufacturing operations (IFR, 2015). Service robots are being used in households, hospital, nursing homes, workplaces, etc. For example, household robots can take over household chores, personal assistant robots can help elderly care or handicapped and surgical robots can improve the accuracy in surgical procedures.

Household robots require knowledge about household objects and their properties to implement tasks given in the commands. There are techniques available for developing knowledge of objects that exist in a ***static*** manner in household environments, such as KNOWROB (Tenorth and Beetz, 2009) and ORO (Lemaignan et al., 2010). However, household environments where the robots work can be dynamic. This is because robots and their human users share the same space and humans can introduce or remove objects into or from the space without informing the working robots. This means that the robots need to continuously develop their knowledge about objects and the properties of the objects.

Ontology is one of the five basic assumptions for knowledge representation (Davis et al., 1993). Ontology specifies knowledge- representation structure in a knowledge base and enables knowledge sharing. For household robots, the ontology specifies what individual household objects should be represented and how the objects are related with one another, and it enables knowledge sharing between different sub-domains, such as sharing common understanding of the context information between users, devices and services in smart home environment (Gu et al., 2004), agents and services in pervasive computing environment (Wang et al., 2004), and between the robots and humans (Haidegger et al., 2013).

Ontology development is an iterative and dynamic process (Gargouri, 2010). Ontologies are widely used in robotic applications (Paull et al., 2012). For example, the RoboEarth European project (Waibel et al., 2011), the Proteus project (Martinet and Patin, 2008), and the A3ME ontology (Herzog et al., 2011). Ontology presents the domain knowledge to be transferred and shared between different groups of humans, robots and other devices. Ontologies have been benefiting robotics to describe and to define concepts, properties and relations between concepts. They share the concepts between robots or between humans and robots.

The knowledge base development techniques use ontologies which are created by domain experts (Tenorth, 2011, Lemaignan et al., 2010, Suh et al., 2007). These ontologies, known as manual ontologies, are manually encoded before robot deployment. Within the household dynamic environments, the manual ontology can lead to situations where the robots are not able to recognise objects that do not exist in the predefined and static ontology and, hence, are not able to understand human user commands that are related to these objects. To support the robots to continuously develop their knowledge, it is necessary to develop techniques that enable the robots to automatically generate ontology, known as ***dynamic ontology***.

In general, a dynamic ontology faces three challenges. The first is to understand objects that are unseen before. The second is to create an ontology with limited human involvement. The third challenge refers to associating semantic knowledge in order to represent descriptions of instances of concepts, properties and values.

The common practice (Noy and McGuinness, 2001) for enabling robots to understand objects that are unseen before is to facilitate the robots to learn about the objects from all available information resources (Tenorth et al., 2012). This involves the identification of the most relevant information and knowledge from various sources, the collection of the information, the definition of new concepts and the relations between the new and the existing concepts and keeping the ontology manageable.

Methods for creating ontology components can generally be classified into statistics-based, linguistics-based and logic-based. Statistics-based methods for creating concepts focus on the identification of lexicons and on the measurement of the co-occurrence between lexicon units, with information obtained through an information retrieval process. Statistics-based methods for creating relations rely on clustering concepts into groups to construct a hierarchy.

Linguistics-based methods for creating concept focus on semantic lexicons. Concepts and relations between concepts are retrieved from the collections of predefined concepts and relations. Linguistics methods for creating relations point to syntactic structure analysis.

Logic-based methods for creating ontology focus on inductive logic programming and logical inference. They have connections with advances in knowledge representation, relation creation and axiom creation. Rules in inductive logic programming are derived from existing collections of concepts and relations. Relations in logical inference are derived from existing collections using rules.

In this research, a combination method of linguistics-based and statistics-based methods is created for concept creation. First, a linguistics-based method is used to create concepts. If a concept cannot be found by the linguistics-based method, then a statistics-based method is applied to create concepts. For creating relations, this research relies on information retrieval due to the availability of a large number of predefined relations between concepts.

In existing works, relation creation focuses on the “is-a” relation. This research, in addition to the “is-a” relation, also creates “HasA” and “MadeOf” relations. The “HasA” and “MadeOf” relations are useful for service robots. The “HasA” and “MadeOf” relations indicate the meaning between two concepts. The “HasA” relation gives the “has” meaning between two concepts and the “MadeOf” relation gives the “made of” meaning between two concepts. They can be used to create a relation between two concepts. Similar objects can be found by searching the “HasA” relation and “MadeOf” relation between concepts.

In existing works, semantic knowledge about instances is not considered. This research also associates semantic knowledge, such as location and time, with instances in the robot ontology. Robots need knowledge about instances to perform tasks and to share between human users and robots in the dynamic environment.

1.2 Aim and Objectives

The aim of this research is to develop a systematic architecture and algorithm for domestic robots to automatically create the ontology components and to associate the semantic knowledge of physical objects in dynamic environments. The objectives are:

- To identify the most suitable keywords in finding out category names
- To create more kinds of relation such as “HasA”, “MadeOf”
- To include semantic knowledge into robot ontology
- To integrate concept component creation and semantic knowledge into a robot ontology framework.

1.3 Research Methodology

In order to understand the state of the art of dynamic ontology for service robots, a literature review is conducted. Research papers are selected from the following areas: methods of ontology development for service robots, ontology learning from text, semantic knowledge and evaluation of ontology learning.

This research applied the ontology learning from text technique in order to automatically build the ontology for service robots in a dynamic environment. The bottom-up strategy is implemented to identify the most specific concepts and to generalise them into more abstract concepts in order to create a concept hierarchy and the relations between concepts by using WordNet (Miller, 1995), ConceptNet 5 (2015) and Web Documents (Embley et al., 1999).

The data and information retrieval method is implemented in order to retrieve the category name for the unknown physical object name and to categorise it into the concept hierarchy by using Web Documents and WordNet. The category name of the unknown object name becomes a concept in WordNet for building the concept hierarchy into the robot ontology.

The method for associating semantic knowledge represents the surrogates for things in the real world. The properties and their property values represent semantic knowledge of the physical objects for sharing semantic knowledge between human users and robots in the environment.

The evaluation method used to query and evaluate the ontology is presented in order to assess the structure of ontology before applying it to robot applications.

1.3.1 Assumptions

This thesis assumes that all physical object names and their property values acquired from the robot recognition are stored in text format. The methods for handling the dynamic ontology consist of the ontology learning from text by using WordNet, ConceptNet and Web Documents for creating the ontology components. WordNet provides a lexical database for the English language. It is suitable for this work because it provides synonyms to create taxonomic relations for constructing the concept hierarchy. The “HasA” and “MadeOf” relations are created between a newly created concept and an existing concept in order to afford the semantic meaning by using ConceptNet. Furthermore, Web Documents provides data to solve the problem of the unknown object names which do not appear in WordNet. In order to learn the unknown physical object names, the category name is retrieved. The semantic knowledge associates with instances of concept. The correction of the robot ontology is assessed by using query and ontology evaluation in order to evaluate the concept hierarchy in the robot ontology.

1.3.2 Proposed Experiments for Assumption Testing

In order to test the hypothesis, a series of experiments are designed in this research. The experiment on creating ontology components presents the concept hierarchy of concepts, relations between concepts, instances of concepts, properties and values. The experiment is to associate semantic knowledge of instances. Before using ontologies in a robot application, the ontology is evaluated to guarantee the quality of its contents.

1.3.3 Scope of the Research

The research focuses on creating algorithms for constructing a dynamic ontology for service robots. This research assumes that all physical object names and property values are retrieved in text format. Text should become a valuable source for creating ontology components and associating semantic knowledge for service robots. The research considers the physical object names in a household environment. The research focuses on handling the process of learning and building an ontology from text for service robots.

This framework focuses on noun, proper noun and adjective parts of speech from text labels of physical objects. The framework cannot solve the problem of the physical object with mismatched labels or which are unlabelled. The ontology learning from text identifies terms, concepts, taxonomic relations and non-taxonomic relations from textual information. Terms are object names and property values are text from input process. Concepts are physical entities in WordNet.

Taxonomic relations are to construct a concept hierarchy ("is-a" relation) in WordNet. Non-taxonomic relations are the interactions between concepts ("HasA" relation and "MadeOf" relation) in ConceptNet. The concept hierarchy and concepts are created and taxonomic relations, "is-a" relation, are retrieved hypernyms from WordNet in order to create taxonomic relations between concepts. The semantic relation is obtained from ConceptNet in order to create non-taxonomic relations.

The category name is retrieved from Web Documents. This category name must only be a physical entity concept in WordNet. The semantic knowledge associates after the concept hierarchy and concepts are built in an ontology. It represents the real- world objects as instances of the concept.

Similar physical objects are recognised in different scenes. This framework cannot tackle the uniqueness of that particular object. For example, there are two water bottles (bottle1, bottle2) in the same scene but they occur in different locations. In the next recognition process, there is one bottle in the environment. This framework cannot guarantee that it is the same bottle: bottle1 or bottle2. It assumes that this is a new instance of bottle in the environment at that time.

1.4 Thesis Organisation

Chapter 1 explains the rationale for the current study, the research setting and the thesis structure. The rest of this thesis is organised into six chapters.

Chapter 2 is a literature review on the state of the art of related works and describes the research methods which are chosen for this study, and explains the reason behind this choice.

Chapter 3 presents the Data-Information Retrieval based Automated Ontology Framework. It consists of seven main modules. There are two phases: the automated ontology and the use of the automated ontology. The automated ontology involves four modules: Data Input, Automatic Ontology, Data and Information Retrieval and Semantic Knowledge Acquisition. The use of the automated ontology consists of three modules: Query, Robot Ontology and Result Evaluation. This chapter also explains the framework design which is applied to this study.

Chapter 4 presents the Automatic Ontology Process. Automatic Ontology consists of five modules: Concept Creation, Relation Creation, OWL Creation, Instance Creation and Property Creation.

Chapter 5 describes the Semantic Knowledge Acquisition Process. The instances of concepts, properties and values are represented in the dynamic environment in order to provide and to share semantic knowledge between service robots and humans.

Chapter 6 presents the Query processes, Result Evaluation processes and system validation. These are used to query and evaluate the ontology in order to assess the content of the ontology. Moreover, system validation provides the validation of the framework via the experiments.

Chapter 7 presents the final conclusions and contributions. Further work is also proposed in this chapter, with respect to discussions on the limitations.

CHAPTER 2 LITERATURE REVIEW

This chapter provides a literature review, the method of development of an ontology for a service robot, including the manual construction methods (Section 2.1.1), the existing ontology methods (Section 2.1.2) and the ontology learning methods (Section 2.1.3). The next section is ontology learning from text which consists of three techniques: statistics-based (Section 2.2.1), linguistics-based (Section 2.2.2) and inductive logic programming (Section 2.2.3). The semantic knowledge (Section 2.3) presents the previous work and the results. Finally, the evaluation of ontology (Section 2.4) indicates the practical definition of the existing evaluation approaches.

2.1 Methods of Ontology Development

In this section, the methods for development of ontologies are discussed and compared. The methods of ontology construction can be classified into three methods: manual construction methods, existing ontology reusing methods and ontology learning methods (Yu and Shen, 2013).

2.1.1 Manual Construction Methods

In this section, the manual methods for the development of an ontology are discussed and compared. The first sub-section gives the general purpose practical manual method. The latter sub-section provides the detail of the specific method for a robot.

The definition of ontology is taken from Information Science. Gruber's definition of ontology as a formal explicit specification of a shared conceptualization (Gruber, 1993) has been utilised by many researchers. Shared conceptualization is commonly accepted understanding for an abstraction of the real world. A formal specification means the machine readability of computational semantics. Explicit definitions of concepts, relations and constraints are distinctly given. Ontology is a study of existence. An ontological commitment is the category of things as determined by the designer or knowledge engineer (Sowa, 2000).

In general, the classical methodologies which built ontologies from scratch are called manual construction. Noy and McGuinness (2001) point out that there are seven processes of iterative design of the ontology lifecycle. To start with, the domain and scope of ontology are determined. The second process considers the reusing of existing ontologies. The next is defining the important terms in ontology. The classes and the class hierarchy are defined in the next step. After that, the properties of classes are defined. The sixth process defines the facets of slots. The final process creates the instances.

The Cyc (Lenat et al., 1990) method performed manual coding of articles and pieces of knowledge by hand. There are two tasks in the Cyc method: (1) the development of a knowledge representation and top-level ontology containing the abstract concepts; (2) the representation of the knowledge of different domains for building the Cyc KB (knowledge base). Cyc presents the consensus knowledge about the world. It is too large and lacks domain-specific knowledge.

Uschold and King’s (1995) method accomplished four tasks. Identifying the purpose of ontology is the first task. After that, building the ontology, evaluating the ontology and writing the document of that ontology. For the purpose of identifying the concepts in an ontology, Uschold (1996) proposed three methods to identify the concepts: bottom-up method, top-down method and middle-out method. The bottom-up method identifies the most specific concepts and then generalises to the general concepts. The most general concepts are identified and the specific concepts are specified in the top-down method. The middle-out strategy identifies the core basic term, then specifying and generalising the concepts as necessary. The bottom-up strategy has been chosen as the method for building the concept hierarchy in the Automatic Ontology process.

This sub-section provides the detail of specific methods for a robot: KNOWROB (Tenorth and Beetz, 2009), ORO (Lemaignan et al., 2010) and the ontology-based unified robot knowledge framework (Suh et al., 2007).

The basic KNOWROB ontology is the manual construction mode with encyclopaedic knowledge about the household domain for a robot. It was adopted from OpenCyc (Lenat, 1995) which is extended Cyc. Therefore, it was manual coding by humans that is not suitable for a dynamic environment.

It can be noted that the Open Robots Ontology (ORO) is designed as a domain-specific common-sense ontology for a service robot. The knowledge storage service can be represented by Resource Description Framework (RDF) and Web Ontology Language (OWL). A new agent is identified by the ORO server and the ORO server automatically creates a new, separate, in-memory OWL model. There are no methods for large-scale knowledge acquisition presented for creating knowledge in ORO.

The knowledge is encoded manually by humans. It is not addressed in a learning system. Ontology-based unified robot knowledge framework is a Prolog-based knowledge representation modelling objects and perceptual concepts as well as actions and situations. It consists of two parts: the knowledge description and the knowledge association. The knowledge description assigns Spaces, Contexts, Objects and Actions, and features knowledge classes for robot data and environments. The knowledge association constructs the relationships between knowledge descriptions. It uses logical inference, Bayesian inference and heuristics method. The knowledge manager handles all requests about the ontology including ontology creation, retrieval, and manipulation. After model knowledge description and knowledge association, a robot ontology was designed using OWL with the Protégé ontology editor (Lim et al., 2011).

A major problem of this framework as shown in the above is the manual creation process. It can be seen that, manually constructed ontology is a time-consuming task and usually requires a domain expert to declare all domain concepts, concept hierarchy and the relations between concepts.

2.1.2 Existing Ontology Reuse Methods

In this section, the existing ontology reuse methods known as ontology re-engineering methods are discussed. Ontology re-engineering is the process of retrieving and mapping a conceptual model of an implemented ontology to a more suitable conceptual model which is re-implemented. There are three activities of re-engineering: reverse engineering, engineering restructuring and forward engineering.

The objective of reverse engineering is to derive the ontology conceptual model from its implementation code. The set of intermediate representations proposed by the extension of METHONTOLOGY method (López et al., 1999) is used to build a conceptual model. The restructuring aims to reorganise the initial concept model into a new concept model. It consists of two phases: analysis and syntactic. The general purpose of the analysis phase is to evaluate the ontology from a technical point of view. The synthesis phase tries to correct the ontology after the analysis phase. This phase also relates to configuration management, which keeps a record of the ontology evaluation and strict change control. Finally, forward engineering aims to output a new ontology implementation on the basis of the new conceptual model.

There have been efforts to reuse existing ontologies. This needs ontology mapping and other methods to match the different standards of ontologies. The METHONTOLOGY method is an example of existing ontology reuse. It executes the candidate ontologies to be reused by emphasizing the ontology components: concepts, properties, relations, constants, formal axioms, rules and instances. Tasks of the conceptualization activity of METHONTOLOGY involve eleven tasks: creating the glossary of terms, creating the concept taxonomies, creating ad hoc binary relation diagrams, creating concept dictionary, describing ad hoc binary relation, describing instance attributes, describing class attributes, describing constants, describing formal axioms, describing rules and describing instances. However, this method retains the problem of a manually constructed ontology. Moreover, the different standards of ontologies need a process of mapping and merging that makes this task complex and time-consuming.

2.1.3 Ontology Learning Methods

The major problems of a manually constructed ontology are it being time-consuming and requiring a domain expert to declare all domain features. Ontology learning is considered an important step in an ontology development cycle. It presents an automated or a semi-automated process of ontology development. The following discussion shows the systems that have been implemented by automatic and semi-automatic process of ontology development. The ontological elements are extracted automatically from different data sources (Buitelaar and Cimiano, 2008). Ontology learning methods can be divided into four approaches: ontology learning from texts, ontology learning from instances, ontology learning from schemata and ontology learning for interoperability (Maedche and Staab, 2000). Ontology learning from texts is applied in this research, as described in section 2.2.

The robot knowledge processing framework, known as KNOWROB includes robot control programs and reasoning methods which connect to the robot's perception and action systems. The knowledge acquisition from the Web in the KNOWROB ontology is a semi-automatic process that created the information about objects from the www.germandeli.com website. It provided an ontology of more than 7,000 object classes and information about objects such as weight, price, country of origin, perishable(ity) and heat sensitive(ity) (Tenorth, 2011).

The purpose is to solve the main problems of the abstract concepts. The abstract concepts are not linked to the robot's perception and actuation systems. Moreover, the observations are combined with encyclopaedic and common-sense knowledge that are based on state-of-the-art semantic web technology in order to reuse existing sources of knowledge. The knowledge processing system presents classes and properties. They are represented in the OWL using Prolog predicates. The class level of modelling in description logics contains abstract terminological knowledge, organised in a taxonomic structure. The instance level of modelling represents concrete and physical objects. The encyclopaedic knowledge models classes of things in the environment and provides the general categories the robot. The structure and some of the concepts are inspired by the Cyc ontology (Tenorth and Beetz, 2013).

The DYNAMO-MAS tool modifies the existing ontology from text in order to reduce the need for human intervention (Sellami et al., 2013). It was developed from the DYNAMic Ontology for information retrieval (DYNAMO) project. There are three corpora from different domains: archaeology techniques, automotive diagnosis and software bug reporting. The process consists of four steps: retrieving the new document from a corpus, DYNAMO Corpus Analyser, DYNAMO Multi-Agent System, and human-evaluated ontology. The Corpus Analyser implemented the term extractor, a lexical relation generator and a lexical relation selector from the YaTeA system (Aubin and Hamon, 2006). The output shows < T_i, Rel, T_j > triplets where T_i and T_j are candidate terms and Rel presents a lexical relation label. A confidence score (Q, I) is computed for each triplet. Q shows the quality of the relation and I present the number of occurrences of the relation. The relation extractor generated four types of lexical relation: Hypernyms, Meronymy, Synonymy and Transverse relations. The DYNAMO-MAS has two evaluation methods: quality evaluation and performance evaluation. The quality evaluation compared a manual ontology development with an automatic ontology development. The time performance and scalability when new documents are added to the corpus was measured. The results showed the percentage of term proposal appearing in the Artal case-study, the Arkeotek dataset and the Actia dataset as 67%, 68.75% and 16.98%, respectively. The results showed the percentage of concept proposal appearing in the Artal case-study, the Arkeotek dataset and the Actia dataset as 56%, 59.26% and 22.22%, respectively. According to the reported experimental results, Knowledge of DYNAMO-MAS is limited and other knowledge sources should be applied to DYNAMO-MAS to improve the results.

2.2 Ontology Learning from Texts

Ontology learning from texts is based on the use of text corpora. A corpus is a set of texts representative of a domain. It is prepared to be processed by a computer and is accepted by the domain expert (McEnery and Wilson, 2001). Figure 2.1 was proposed by Buitelaar et al. (2005) in order to classify an ontology learning approach according to the task. These tasks consist of term extraction, synonym discovery, concept formation, concept hierarchies, relations and rules.

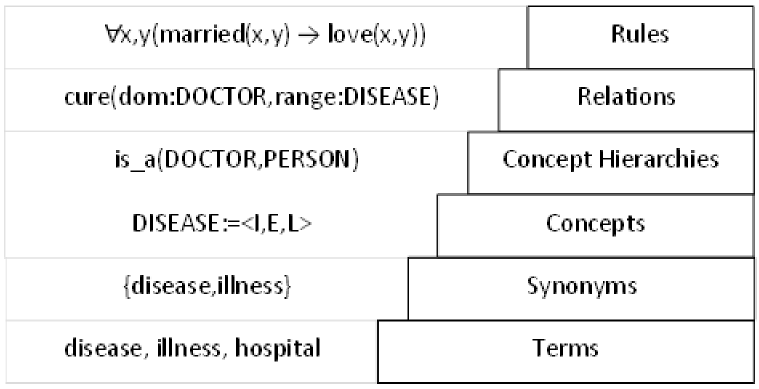


Figure 2.1 Ontology Learning Layer Cake (Buitelaar et al., 2005)

The first task of ontology learning is term extraction, which determines the relevant phrases and terms for specific domain. A term layer is a prerequisite and the first step of ontology learning from text. Term extraction applies the linguistic processing. It uses a part-of-speech (POS) tagger to identify the internal semantic structure over the domain. The second, synonym layer focuses on how to appropriately discover synonyms of terms. Ambiguous terms can appear in a particular domain. The layer integrates WordNet for retrieving the English synonyms and EuroWordNet for getting multilingual synonyms. The next layer, the concept formation task, consists of three parts: concept intension, concept extension and lexical realization. Concept intension presents a description of the concept from a dictionary. Concept extensions show a set of instances of the concept. The lexical realization is the term defining the concept from the corpus.

Buitelaar defined a concept as $(\Im, \Sigma) \oplus L$ where $\Im$ is the intension of the concepts. An intensional definition provides the meaning of a term by specifying all the properties (Keizer et al., 2000). The necessary and sufficient properties are grouped for the set of definition. Σ is a set of instances of a concept. L presents its linguistic realization. The concept learning consists of the derivative of formal and informal definitions. The formal definition might be a textual description, whereas the informal definition obtains the extraction of relations between a particular concept and other concepts. Buitelaar also presented three paradigms for taxonomies from text: the lexico-syntactic pattern application, the Harris distributional hypothesis and the information retrieval community. The lexico-syntactic pattern is used for getting hyponymy relations. It has reasonable precision results but the recall result was very low. The Harris distributional hypothesis presented the context of synonym extraction and term clustering. The information retrieval paradigm focused on a document-based notion of term subsumption (Sanderson and Croft, 1999). The fifth layer is the relations layer. Ontology involves a hierarchy backbone (is-a relation) and non-hierarchical relations. Finally, rules are axiomatic definitions of concepts.

Lin and Pantel (2001) proposed the Extended Distributional Hypothesis. If the paths in dependency trees have similar meaning, they are liable to connect similar sets of words. The algorithm generates inference rules by finding similar paths. The learning and construction of complex axioms approach presented three modules for symbolically obtaining axioms from text (Ribeiro et al., 2014). The syntactic parsing module applied Probabilistic Context-free Grammar for analysing the sentences. The semantic parsing module consists of four activities: term extraction, concatenation, phrase breaking and relations extraction in order to detect the terms and relations between them. The hardest part of the ontology creation process is the OWL DL Axioms module. The first step is construction of taxonomic relations: the pattern <NPs> <VP> <NPs> is applied where <VP> is a verb (is a/an, is or are). Next, construction of non-hierarchical relations implemented <NPs> <VP> <NPs> pattern where <VP> is a verb other than (is a/an, is or are). The verification of conjunctions (and) and disjunctions (or) is required in order to verify and analyse the pattern. The last step is detection of negations. The two hierarchical axioms, one union between concepts (unionOf) and one negation of properties (complementOf) are produced as axioms. The result shows that need of automatic creation of expressive axioms is adequate to create ontologies with Attributive Concept Language with Complements (ALC) expressivity.

Al-Arfaj and Al-Salman (2015) classified ontology learning approaches according to several main dimensions: type of knowledge resources, level of automation, learning targets, purpose and learning techniques. There are three types of knowledge resource from which to learn an ontology: structured, semi-structured and unstructured data. Structured data is defined knowledge models including existing ontologies and database schema. Semi-structured data is related to the mixed structured data with free text such as Web pages, Wikipedia, dictionaries and XML documents. Finally, textual content is called unstructured data. There are two levels of automation: semi-automation with user intervention and full automation without user intervention. The learning targets describe the concepts and axioms which identify the criteria of concepts and relations. The purpose of ontology learning can be created from scratch or by updating an existing ontology.

There are several approaches for the partial automation process such as natural language analysis and machine learning technique. Maedche and Staab (2000) classify the ontology approaches as: ontology learning from texts, ontology learning from instances, ontology learning from schemata and ontology learning for interoperability. This research focuses on handling the process of ontology learning from text.

Ontology learning from text (Wong et al., 2012) is an automatic or a semi-automatic process of ontology construction and maintenance, identifying terms, concepts, relations, and axioms from textual information. There are five types of output in ontology learning: terms, concepts, taxonomic relations, non-taxonomic relations and axioms. Terms are defined as lexical realization which are single word or multi-word and relevant to a domain. Concepts are created by grouping similar terms. A taxonomic relation organises concepts into a hierarchy. Non-taxonomic relations are the interactions between the concepts. Axioms are sentences that examine and define the correctness of the ontology.

An automatic or semi-automatic tool for ontology construction has not yet reached the goal of fully automating the ontology development process over the past years (Barforush and Rahnama, 2012). Barforush and Rahnama pointed out three types of input used by a learning process: ontology learning from structured data, semi-structured data and unstructured data. An ontology learning process extracts the structured information from sources such as database schemas, existing ontologies and knowledge bases. The semi-structured data provides the semantic information, such as WorldNet (Miller, 1995), HTML and XML documents. Most of the available knowledge is in the form of unstructured data for learning input. Examples of unstructured data are natural language texts, word documents and text documents. An ontology creation method consists of concept learning and taxonomy construction, and identifying non-taxonomic relation.

The first objective of concept learning and taxonomy construction is to retrieve terms and create a hierarchy. There are three learning process approaches: document-based, synonym extraction and pattern-based. The document-based approach focuses on concept formation (Sanderson and Croft, 1999). Synonym extraction presents terms which share similar syntactic contexts (Bisson et al., 2000; Caraballo et al., 1999). The pattern-based approach is a heuristic method using regular expressions to find taxonomic relations expressed in texts, for example, Hearst patterns (Hearst, 1992; Auger and Barrière, 2008).

The second objective of concept learning and taxonomy construction is to construct a taxonomy of concepts using *is-a* relations, for example, clustering (Bisson et al., 2000), WordNet-based, lexico-syntactic pattern (Maynard et al., 2009) and statistical analysis (Suchanek et al., 2006). Identifying non-taxonomic relations is the task to detect related concepts and consider how these concepts are related. Examples of learning relationship from text depend on the degree of generality of the relation extraction. Berland and Charniak (1999) purposed the *subtype-supertype* relations (*part-of*) learning approach.

The RelExt is an automatic approach identifying a pair of concepts connected by a relation from an existing ontology (Schutz and Buitelaar, 2005). This work has not been applied to axiom and rule learning. Examples of axiom and rule learning are: Völker proposed the method for automatically building complex class descriptions (Völker et al., 2007) and taxonomy refinement (Völker and Rudolph, 2008). Automatic approaches for axiom creation on WordNet have been presented by Navigli and Velardi (2006), Navigli and Velardi (2008) and Moldovan et al. (2007).

Barforush and Ali (2012) classified the new ontology creation methods into five methods: an iterative view, a multilingual view, a web- based knowledge acquisition view, a process engineering view and a design pattern view. An iterative view involves three resources: a corpus of texts, a set of lexico-syntactic and a set of RDF triples (Brewster et al., 2007). Hjelm (2009) proposed a multilingual view method in order to improve the robustness and predictability of evaluation measurement. This method merged information across different languages and presented an automatic evaluation of a learning ontology.

Sánchez (2009) presented a web-based knowledge acquisition approach from the web which consists of automatic, unsupervised and domain-independent techniques. The main processes of the method are the extraction and selection of related terms, the taxonomic organisation, the non-taxonomical relationship labelling and the named entities, class features detection. A process engineering view (Tempich et al., 2008) presented the ontology learning process in terms of activities, actor, inputs, outputs and support tools. The method consists of eight processes: feasibility study, requirements specification, selection of information source and ontology learning, learning preparation, learning execution, ontology evaluation and ontology integration (Gangemi, 2005).

Small ontologies connect between problem types and design solutions (Presutti and Gangemi, 2008). Barforush and Rahnama (2012) also proposed the extended framework dimension, sub-dimensions and values. This framework consists of six sub-dimensions: element learning, starting point, learning method, pre-processing, the result and evaluation. Element learning is classified into three elements: concept, relation and axiom. Instances are sub-classes of concept and relation has two types: taxonomic and non-taxonomic. The second dimension is starting point: prior knowledge and input. Prior knowledge consists of ontology and lexicon.

The input type is divided into structured data, semi-structured data and unstructured data. The learning method dimension consists of category, task and degree of automation. The learning method category has supervised/unsupervised and online/offline. The learning method approach is classified into statistic approach, logical approach, linguistic-based, pattern/template matching and combined approach. The learning method task consists of classification, clustering, rule learning, concept formation and ontology population. Degree of automation can be automatic and semi-automatic or cooperative, as shown in Figure 2.2.

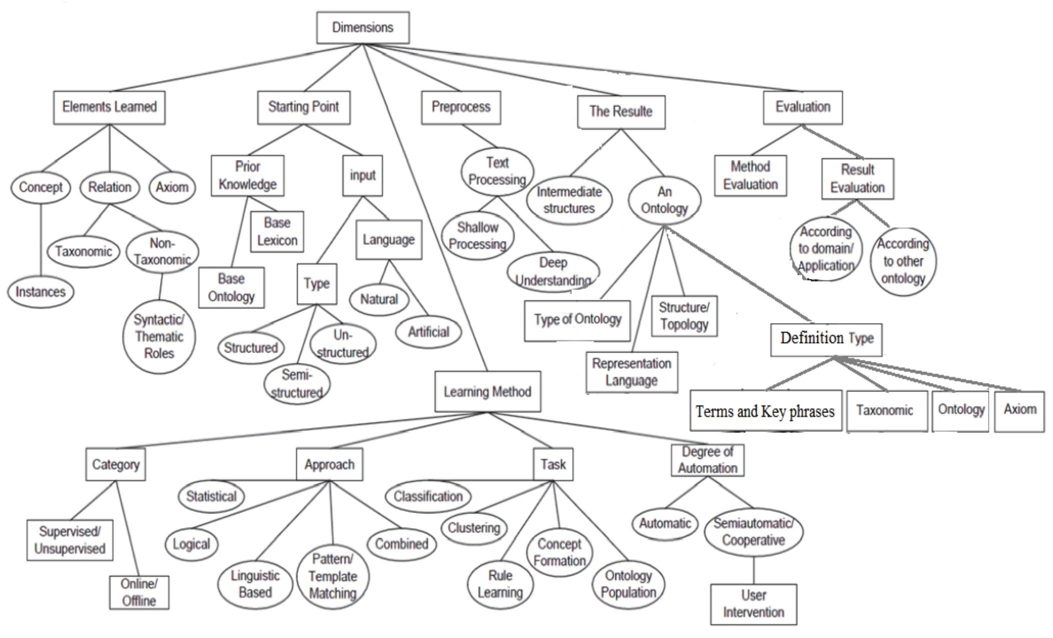


Figure 2.2 Framework dimension (Barforush and Ali, 2012)

Gacitua (2008) proposed OntoLance, which is a framework for an ontology learning technique. This framework can be classified as an automatic method of ontology learning from unstructured text and the output structure is an ontology.

Wong et al. (2012) classified ontology learning techniques into statistics-based (Section 2.2.1), linguistics-based (Section 2.2.2) and logic-based (Section 2.2.3) based on the tasks to be accomplished, as given in Table 2.1 which compares methods of ontology learning from texts.

2.2.1 Statistics-based Techniques

The statistics-based techniques are derived from information retrieval, machine learning and data mining. Syntagmatic similarity is the association between terms. There are two types of syntagmatic similarity, namely, semantic similarity and semantic relatedness. The concept of semantic similarity is more specific than semantic relatedness and measures the degree to which two concepts are similar. Two concepts are connected through hierarchical is-a relations (Wong et al., 2012).

The techniques of machine learning and statistical natural language processing were applied to construct the domain ontology semi-automatically and extract domain concepts from a corpus. The acquisition of domain concepts includes extracting terminology from texts, synonym recognition and domain concept selection using an n-gram approach and matching method to retrieve synonyms from professional dictionaries. The improved algorithm of hierarchy clustering constructed the hierarchy relationships and the criterion of high cohesiveness and low coupling is applied in the measure of clustering results. The natural language processing is utilised to extract subject, predicate and object of sentences from a Chinese corpus. Finally, the system is implemented by Jena API interface (an open source Semantic Web framework for Java language). The results showed that the revised hierarchy construction algorithm decreases the depth of clustering and increases the leaves of nodes. However, the class caption cannot be automatically named in the hierarchy relation reorganisation and cannot extract logical relations from a corpus (He and Hou, 2008).

The framework of a semi-automatic domain ontology system (Dan et al., 2010) consists of three parts: the extraction module of domain concepts and the extraction module of taxonomy and non-taxonomy relations among domain concepts. The domain concept extraction applied a statistical analysis method in order to extract compound words from the Chinese Lexical Analysis System developed by the Institute of Computing Technology, Chinese Academy of Sciences. Moreover, there are three parameters which validate the compound words consisting of mutual information, context dependency analysis and domain relativity analysis. The extraction of relations is implemented by generalised suffix tree and clustering. The algorithm based on association rules mining is applied to extract related concept pairs. The results showed compound and common words from the extraction process. However, this data source was chosen manually and the extracted compound words were not use characteristics of the language.

2.2.2 Linguistics-based Techniques

The linguistics-based techniques depend on natural language processing. Some of the techniques include semantic lexicons, lexico-syntactic and part-of-speech tagging. The semantic lexicons implement a large collection of predefined concepts and relations, such as WordNet (Wong et al., 2012).

This semantic robot service (Ukai et al., 2009) integrates multiple ontologies, user request, robot service, robot function, robot structure, object, insertion task and recovery task ontologies. The description language of the ontology used OWL to describe the four purposes of ontology: Ontology for User Objectives, Ontology for User Methods, Ontology for Tools and Ontology for Instances of Tools. The Ontology for User Objectives describes the relation between objects and user actions. The Ontology for User Methods is used for explaining the relation between user actions and user primitive actions. The Ontology for Tools shows the relation between user primitive actions and concept tools. The Ontology for Instances of Tools depicts the relation between concept tools and instance tools in the actual rooms.

Table 2.1 Comparison of methods of ontology learning from texts

Method	**Research**	**Method to create concepts and relations**	**Disadvantage**
Statistics-based	He and Hou, 2008	The domain concept selection uses an n-gram approach and matching method to retrieve synonyms from professional dictionaries. The construction of hierarchy relations uses the relevancy algorithm to calculate the possibility of the occurrence for building the hierarchy system of related concepts.	The class caption cannot be automatically named in the hierarchy relation reorganisation. System cannot extract logical relations from a corpus.
	Dan et al., 2010	Statistical analysis method creates a suffix tree. Clustering and association rule mining are adopted in domain concept extraction and relation extraction in order to extract compound words.	This data source was chosen manually and the extracted compound words are not use characteristics of the language.
	Sun et al., 2014	A set of appearance attributes and name classifiers are learned and arranged as a tree hierarchy by the vector space. The system knows all object categories and identifies objects into a tree hierarchy.	Ontology is not created in this work. It does not add new categories. The new name needs to associate with the existing name.

Linguistics-based	Ukai et al., 2009 and Ngo et al., 2010	This approach used search engines to build a corpus for RT ontology by lightweight NLP techniques. The learning algorithm was divided into three steps: term extraction, task candidate selection and RT ontology creation. The RT ontology extracts knowledge by using the grammatical relation between verb (relation) and noun (concept) in sentences from English books, ConceptNet and Google 1T 5-gram.	Data source is limited. The system cannot create new concepts that do not exist in the specific data sources.
	Zhou et al. ,2006	A core ontology built by a human expert and ontology extension by WordNet. The event-based learning method obtains an event (c1, v, c2) with two ontology concepts (c1, c2) and one relation (v).	Human builds the core ontology.
Logic-based	D'Este and Sammut, 2008	A method to learn semantic knowledge via dialogues by using inductive logic programming in order to build an ontology. The learning system utilised a Horn clause for creating a concept. The relationships between objects must be specified by a walking technique.	The robot satisfies the conditions in the concept description and requests feedback from the user.

The robot technology (RT) ontology learning from text (Ngo et al., 2010) consists of a corpus builder and information extraction. The environment description file is an XML structure which consists of Object ID attribute and Object name attribute as the symbolic representation of the object in the local data server. This approach used search engines to build a corpus for RT ontology. The learning algorithm was divided into three steps: term extraction by lightweight (taxonomic hierarchy and properties between concepts) techniques, task candidate selection and RT ontology creation. The term extraction by lightweight NLP techniques considers four sentence structures which show the relationship between two nouns and extract verb phrases using part-of-speech tagging technique. The object-task map between objects and tasks was created in the task candidate selection. A Jena framework was implemented to explain the object-task map to the RT ontology. The RT ontology is the automatic approach based on basic-level knowledge (Ngo et al., 2011). It consists of a Where layer, a What layer and a How layer based on 4W1H (Where, When, What, Who and How) and three classes of semantic concepts. It extracts objects and human activities from education books and MIT's ConceptNet. The automatic knowledge retrieval consists of: object extraction process, activity extraction process and connection extraction process. The RT ontology extracts knowledge by using the grammatical relation between verb and noun in sentence. The "Bring something" robot service (Lam et al., 2012) represented common- sense knowledge to build an RT ontology in order to learn new knowledge and generate robot services. For Place-Object, Place-Activity and Object-Activity connections, weighting vector (W) is defined as equation 2.1.

$$W = (W_b, W_c, W_g, W_u) \tag{2.1}$$

For Object-Object connection, weighting vector (W_{oo}) is defined with one more parameter as equation 2.2.

$$W_{oo} = (W_b, W_c, W_g, W_n, W_u) \tag{2.2}$$

W_b is frequency from books, W_c is reliability score from ConceptNet, W_g is frequency from Google 1T 5-gram, W_n is similarity score of two objects based on

WordNet and W_u is user evaluation score. The connection (n_1, n_2) between two concepts n_1 and n_2, W_b is defined as per the following. $W_b = f(n_1, n_2)$, the number of times that the connection (n_1, n_2) is extracted from books. W_c = score (n_1, n_2), the reliability score of the assertion between n_1 and n_2 from ConceptNet. $W_g = f_{gram}(n_1, n_2)$, the frequency of the n-grams that contain n_1 and n_2 from the Google Web 1T 5-gram (Brants and Franz, 2006).

An n-gram corpus was generated from a source of approximately one trillion words collected by Google. The web service interface of Web1T5-Easy (Evert, 2010) calculated the f_{gram}. W_u is the user evaluation score. This parameter is left for future development of the RT ontology. $W_n = d_{WordNet}(n_1, n_2)$, the distance between two nouns n_1 and n_2. It is calculated from WordNet hypernyms.

The RT ontology quality is evaluated by manually judging two main connections: Place-Object and Object-Activity. The Place-Object connection considered the object usually appears in the corresponding place in normal condition as the common objects. The results showed the percentage of common objects appearing in kitchen, living room and bedroom by 86%, 93% and 64 %, respectively. The Object-Activity connection considered the activity can be conducted with the object in real life as the relevant activities. The results showed the percentage of relevant activities appearing in kitchen, living room and bedroom by 60%, 57% and 39%, respectively. It can be seen that the manual evaluation was performed based on human common sense. Educational books are appropriate to provide basic-level knowledge, but the number of data is limited.

For the purpose of the automatic ontology creation, the concept hierarchy creation is the principal process that should be accomplished. The concept hierarchy creation in the Automatic Taxonomy Construction from Text framework (ATCT) (Meijer et al., 2014) proposed the method to construct the broader-narrower relations between concepts by calculating the co-occurrence of different concepts (Sanderson and Croft, 1999).

$$P(x|y) \geq t, P(y|x) < t \tag{2.3}$$

In equation 2.3, t is a co-occurrence threshold value. If concept x shows in at least the proportion t of all documents in which concept y shows and if concept y shows in less than the proportion t of all documents in which concept x shows, then concept x is a parent concept of concept y. A subclass of the subsuming concept is a concept subsumed by another concept. The subsumption relation creates a hierarchy of classes. The subsumption relation is the inheritance of properties from the parent concept (subsuming) to the child concept (subsumed).

For parent selection, this work proposed equation 2.4:

$$score(p,x) = P(p|x) + \sum_{a \in A_p} P(a|x) \qquad (2.4)$$

Where p is a parent concept of concept x, A_p is a list of ancestors of p and w (a, x) is a weight value with which the conditional probability P (a | x) of ancestor a given x is multiplied. The weight value is as equation 2.5.

$$w(a,x) = \frac{1}{d(a,x)} \qquad (2.5)$$

d (a, x) is the distance between node x and ancestor node a. The processing speed integrates with the ability for providing a good concept broader-narrower relation. As a result, this method uses a short time to categorise concepts in a concept hierarchy.

This framework compared the constructed taxonomy with a reference taxonomy using golden standard evaluation approach. It presents the semantic precision (SP) and semantic recall (SR), as equation 2.6.

$$SP(T_C, T_R) = \frac{|C_C \cap C_R|}{|C_C|}$$
$$SR(T_C, T_R) = \frac{|C_C \cap C_R|}{|C_R|} \qquad (2.6)$$

Where T_C is the core (built) taxonomy and T_R is reference taxonomy, C_C is the concepts of the core taxonomy, and C_R is the collection of concepts of the reference taxonomy.

The common semantic cotopy (csc) is the collection of a concept and its parent concepts and child concepts. This collection is shared by a core taxonomy and reference taxonomy in order to calculate the quality of the relations in the built taxonomy (Dellschaft and Staab, 2006).

$$\mathrm{csc}(c, T_C, T_R) = \{c_i | c_i \in C_C \cap C_R \wedge (c_i \leq C_C\ c \vee c\ \leq C_C\ c_i)\} \qquad (2.7)$$

In equation 2.7, c is a concept and C_C is the order of the broader-narrower relations in the TC taxonomy. The global taxonomic precision (TP) and global taxonomic recall (TR) employ the csc to compare the relations of the core taxonomy concepts and reference taxonomy concepts. The local taxonomic precision (tp) and local taxonomic recall (tr) are defined in order to define the TP and TR. The definitions of tp and tr are as equation 2.8.

$$tp_{csc}(c, T_C, T_R) = \frac{|\mathrm{csc}(\mathrm{c}, \mathrm{T}_C, T_R) \cap \mathrm{csc}(c, T_R, T_C)|}{|\mathrm{csc}(c, T_C, T_R)|}$$
$$tr_{csc}(c, T_C, T_R) = \frac{|\mathrm{csc}(\mathrm{c}, \mathrm{T}_C, T_R) \cap \mathrm{csc}(c, T_R, T_C)|}{|\mathrm{csc}(c, T_R, T_C)|} \qquad (2.8)$$

Both tp and tr show the quality of the relations of a single concept. The measure takes the intersection of the csc viewed from the core taxonomy's perspective and from the perspective of the reference taxonomy, respectively. The TP and TR are defined as equation 2.9.

$$TP_{csc}(T_C, T_R) = \frac{1}{|C_C \cap C_R|} \sum_{c \in C_C \cap C_R} tp_{csc}(c, T_C, T_R)$$
$$TR_{csc}(T_C, T_R) = \frac{1}{|C_C \cap C_R|} \sum_{c \in C_C \cap C_R} tr_{csc}(c, T_C, T_R) \qquad (2.9)$$

The taxonomic F-measure (TF) describes the quality of the concept broader-narrower relations. The TF is defined as equation 2.10.

$$TF(T_C, T_R) = \frac{2 \cdot TP_{csc}(T_C, T_R) \cdot TR_{csc}(T_C, T_R)}{TP_{csc}(T_C, T_R) + T_{csc}(T_C, T_R)} \qquad (2.10)$$

This approach built a taxonomy for the domain of economics and management. According to the reported experimental results, semantic precision was 11.63%, semantic recall was 5.57% and the taxonomic F-measure was 68.16%.

Zhou et al. (2006) proposed a semi-automatic ontology learning based on WordNet and event-based natural language processing, a core ontology built by a human expert and ontology extension by WordNet. Next, it applied an event-based learning method to obtain new knowledge from a domain corpus. As a result, the new concepts and relations were added into the ontology. Experimental results reported that there were 539 of four types of concepts: synonymy, hypernym/hyponym, holonym/meronym and domain terms. The new concepts and relations were learned from WordNet and were acquired from event extraction. However, this approach used a human expert to create the core ontology.

The training stage in an identification system for RGB-D scenes (Sun et al., 2014) learned a set of appearance attributes and name classifiers. There are three types of appearance attributes: colour, shape and material. A multinomial logistic regression model is modelled for each attribute type as equation 2.11.

$$P\left(a_t^k \middle| o\right) = \frac{\exp(F_t^k I_o)}{\sum_{t=1}^{T} \exp(F_t^k I_o)} \tag{2.11}$$

T gives the number of attribute values for the *k*-th attribute type. The parameter vector of the linear discriminative function is presented in F^k_t . I_o is the RGB-D feature vector of an "o" object. It is extracted using hierarchical matching pursuit. After learning a set of attribute classifiers and name classifiers, the object name is arranged as a tree hierarchy *H*. In the case where the "w" object name is already contained in a node n^w_l of the tree *H*, then the probability P (w | o) for an "o" object is computed as equation 2.12.

$$P(w|o) = \prod_{l=1}^{L-1} P_{n_l^w}(n_{l+1}^w|o) \tag{2.12}$$

Each $P_{n_l^w}(n_{l+1}^w|o)$ is a probability of node n_{l+1}^w given the RGB-D feature of object o evaluated via the classifier trained for node n_l^w. It assumes that the system knows all object categories; it does not add new categories. The new name needs to associate with the existing name and find a path. Given a new name *w*, if there is a synonym of the new name then the word is added to the node of its synonym. In the case where a hyponym of a leaf node is the new word, then the new word is added as a child of that leaf node.

A new name word w is a d-dimensional vector $v^w \in \mathrm{R}^d$ using vector space model. v^w is given by the co-occurrence with other words within a large document corpus (Socher et al., 2012). The decision function ($\psi : (w, \mathcal{H}, p) \rightarrow \mathbb{R}$) measures the match score between a path p and word w given the hierarchy H as equation 2.13.

$$p^w = \arg\max_{\mathrm{p}} \psi(w, \mathcal{H}, p) \qquad (2.13)$$

p^w is the correct path. A word replaced with vector representation v^w. The decision score of the path as the sum of scores of all nodes in it for a path p with L words as equation 2.14.

$$\psi(w, \mathcal{H}, p) = \sum_{l=1}^{L} \frac{\emptyset(v, v_l)}{L} \qquad (2.14)$$

Where $\emptyset(v, v_l)$ calculates the match score between words v and v_1. The negative distance between v and v_1 is $\emptyset(v, v_l) = -(v - v_l)^T(v - v_l)$. The vector space does not provide the semantic information structure. The decision score function is not increased by the correct path. Experimental results reported that the system achieved 74% identification accuracy for scenes containing six objects, with half of the objects being unknown on average. The learned names have some errors when people use higher level names of an object that appears in the real- world scenes. This work does not build ontology and semantic knowledge of physical objects.

2.2.3 Inductive Logic Programming

There are two logic-based techniques, inductive logic programming and logical inference. The inductive logic programming rules are obtained from existing collections of concepts and relations. There are two types of rule, positive and negative examples. Logical inference is descended from existing ones using rules and the conclusion always follows the stated premises (Wong et al., 2012).

D’Este and Sammut (2008) presented a method to learn semantic knowledge via dialogues by using inductive logic programming in order to build an ontology. The learning system utilised a Horn clause for creating a concept description. The conditions on the right hand side were retrieved from sensor readings. The predicate on the left hand side was derived from human speech. Concept learning is the process of collection of the feature names and values in order to describe the objects. There are six steps required to build up the hypothesis for concepts, such as: obtaining the name of concept, identifying the known colour in the scene, creating a blob of colour, processing the blob and finding the suitable feature values for each of the known features, categorising the feature values and building a clause from the categorised features names. Concept learning consists of the relationships between objects and the generalising from examples. In the relationships between objects process, D’Este and Sammut applied the ability to gain information from a robot for creating relationships between objects. According to the algorithm for creating multiple object concepts, the first step is retrieving the concept name and the colour of the object. Next, constructing a blob of colour is implemented. Third, processing the blob and the suitable feature values for each of the known features is defined. Th next step is executing the blob and searching the suitable feature values for each blob of other known colours, and then calculating the relative positions to the other blobs for each blob. Lastly, a clause is built from the categorised feature names of each blob. The generalising from examples process, when the robot found the unrecognised object, is when D’Este and Sammut match the new object with the current concept and generalises the most suitable concept. The robot satisfies the conditions in the concept description and requests feedback from the user. The experiment results showed the learning performance of D’Este and Sammut’s system, the robot spent the most of time walking around the object for visioning. The three names to be learned consisted of obstruction, inference and plane. The correct results were 87.5%, 75% and 100%, respectively.

2.3 Semantic Knowledge

Semantic knowledge is modelled with expressions. It focuses on the specific aspect of language communication. Linguistic knowledge is expressed by lexical entries. It can be understood and used by humans (Velardi et al., 1991). A robot needs semantic knowledge for the purpose of natural language processing. The conceptual meaning type for the lexicon is the cognitive content of words. It is expressed by features or by primitives. It presents phenomena that are embedded in language. The collocative meaning type for the lexicon describes the incomprehensible words that appear together in everyday language. It does not present the real sense of a word. It clarifies the word associations in terms of meaning relation between a lexical item and other items or classes. The collocative meaning relies on solid evidence represented by word associations and considers the interpretation of an association. The valid associations are a marked phenomenon. Both conceptual meaning and collocative meaning are represented in the natural language processing literature using some subjective, human-produced set of primitives (conceptual dependencies, semantic relations, conceptual categories) (Velardi and Pazienza, 1989).

Recent efforts in human–robot interaction present the semantic knowledge in the robotics area. There are two important aspects: the need for an explicit representation of knowledge and the need for grounding the symbols used in this representation (Hertzberg and Saffiotti, 2008).

The first aspect is explicit representation. The semantic knowledge presents the descriptions of the concepts and relations of the domain. These descriptions are represented explicitly inside the system. Semantic knowledge is used and is presented by a robot. The recognition process aims to attach labels to the sensor data. These labels have to be embedded in a domain. It allows the robot to reason. When the labels are purely syntactic, the robot can utilise the semantic knowledge.

The second aspect is symbol grounding. The robot grounded all elements in the knowledge representation with the robot's sensor and motor signals. Hertzberg also considers three aspects co-exist in a knowledge-based robot. The sensor data consists of 2D or 3D laser data, camera image, or both, which provides the semantic knowledge. The ability to use semantic knowledge aims to improve planning and control aspects. The final aspect is the usage of semantic knowledge. It focuses on methods and concepts in robotics.

The research by Holzapfel et al. presented the learning of new words from speech recognition. The dialogue model is used for acquiring semantic knowledge (Holzapfel et al., 2008). The representational knowledge bases and interaction knowledge are components of knowledge bases. The representational knowledge bases define the aspects of extended knowledge. It acquires new information. The interaction knowledge presents the method to obtain the knowledge via a communication process. This ontology consists of functional concepts, classes and properties. The functional concepts describe how an object class can be used. The classes are arranged in object class hierarchy. In the dialogue system, the classes apply typed feature structures (TFS) to represent semantics (Carpenter, 1992). The deficient information occurs when user input cannot be understood (speech recognition and understanding) and the specific object cannot be found (visual processing of objects).

The Head-Tail model (Schaaf, 2001) was applied for detecting the unknown word and the out-of-vocabulary (OOV) recognised the unknown words. The result shows that, the unknown words occurred at specific positions in the grammar. The research aims to learn object properties and learn an object's category. The learning of object properties detects the unknown property values by using OOV detection. It is added to the dictionary of the speech recogniser and to the speech recognition. The learning of an object's category links a manual category by user input and a prompted mode which creates browsing the ontology. The experiments utilised 52 dialogues from six naïve users. The user did not know the objects were known and unknown to the robot. The system categorised and learned new words, properties and types of object in dialogue with the user. According to the overall results of the experiment and recognition rates of visual object recognition, for all experiments, there were 52 dialogues, 40 unknown objects and 12 known objects. The correct results were 49, 39 and 10, respectively (Bouguerra et al., 2008).

2.4 Evaluation of Ontology Learning

Ontologies specify the knowledge in a standard way between human and robot. Ontologies are engineering artefacts that need to be evaluated before they are deployed in applications. The evaluation of ontologies concerns the assessment of the resulting ontology. The resultant ontology is produced by an ontology learning method. Ontology evaluation aims to ensure that the resulting ontology represents accurately the domain. In general, ontologies are not an end product and are gained for other tasks. Therefore, an ontology evaluation approach is needed in order to decide which produced ontology is suitable for the requirement.

Typically, ontology learning evaluation methods are aimed at evaluating structural and functional aspects. They can be classified into two main evaluating methods: the quality assurance during ontology engineering process and the comparing ontology learning (Dellschaft and Staab, 2008). The first method can be classified into task-based, corpus-based and criteria-based evaluation approaches. The second method can be a manual evaluation by a domain expert or gold standard-based evaluation. The first scenario evaluates during the ontology engineering process which is considered consistent, complete, concise and expandable. The important part of this scenario is the quality assurance process. This scenario consists of three approaches to a functional and structural: task-based, corpus-based and criteria-based.

The first approach is task-based evaluation. It evaluates the adequacy of ontologies in the context of a certain task. The changing constant of evaluation influences the results of the ontology because the evaluation is dependent on the specific task. Second, corpus-based checks of the ontology are adequate to support the given domain. The components of the ontology are compared with the contents of a text corpus. The content of the corpus is analysed with a natural language method. It is suitable for evaluating the ontologies that are created from an ontology learning algorithm. The final approach is criteria-based, measuring ontology or taxonomy with the criteria. The qualitative criteria-based user evaluation consists of five measurements (Chuang et al., 2005). First, cohesiveness measures similarity of the clustered instances in a semantic way. The second measurement is isolation; it tests the level of the auto-generated clusters. Third, measuring the hierarchy, the hierarchy is traversed from broader concepts to narrower concepts. The navigation balance makes a decision on the fan-out at each level of the hierarchy. Finally, readability considers how easy it is to recognise the concepts of clusters at all levels. The cost in terms of time consumption and labour is the main problem of the approach.

The second scenario is manual evaluation by a domain expert and gold standard-based evaluation. On one hand, the manual evaluation by a domain expert approach uses the human expert to judge the correctness of system. There are several disadvantages of this approach. The extracted information is compared with the knowledge of the human expert. The knowledge of a human expert is not the method for measuring the precision of a learning algorithm. Moreover, the factors of the evaluation depend on the expert. It is not suitable for all situations. On the other hand, comparing the previously created gold standard with the learned ontology is the main purposed of the gold standard-based approach. The gold standard is the idealized solution of the learning algorithm. In the case where the learned ontology has a high similarity with the gold standard, this learning algorithm has good results.

Dellschaft and Staab (2008) defined the structure O: = (C, *root*, $c \leq$ C) as a core ontology. C is a set of concept identifiers. *Root* is a root concept for the partial order on a set of concept identifiers. Concept hierarchy or taxonomy is partial order. The equation $\forall c \in C$: $c \leq C$ *root* carries for the concept hierarchy.

Precision and Recall are used for comparing a reference retrieval (*Ref*) with a computed retrieval (*Comp*). They are defined as equations 2.15 and 2.16.

$$P(Ref, Comp) = \frac{|Comp \cap Ref|}{|Comp|} \qquad (2.15)$$

$$R(Ref, Comp) = \frac{|Comp \cap Ref|}{|Ref|} \qquad (2.16)$$

The F1-measure is used for balancing the precision and recall values as equation 2.17.

$$F_1(Ref, Comp) = \frac{2 \cdot P(Ref, Comp) \cdot R(Ref, Comp)}{P(Ref, Comp) + R(Ref, Comp)} \qquad (2.17)$$

A core ontology (O_C), a reference ontology (O_R), lexical precision (LP) and recall precision (RL) are defined as equation 2.18. They aim to evaluate the learned terms. The learned terms should cover the target domain.

$$LP(O_C, O_R) = \frac{|\, C_C \cap C_R|}{|C_C|} \qquad LR(O_C, O_R) = \frac{|\, C_C \cap C_R|}{|C_R|} \qquad (2.18)$$

For evaluating the concept hierarchy, the taxonomic precision and recall are defined for comparing concepts, comparing concept hierarchies and retaining concrete measures. First, comparing concepts compared the concepts and concept hierarchies. It allocates them into the local and global measures. The positions of two concepts are compared by the local measure comparison. On the other hand, the global measure compared two whole concept hierarchies.

$$tp_{ce}(c_1, c_2, O_C, O_R) := \frac{|ce(c_1, O_C) \cap ce(c_2, O_R)|}{|ce(c_1, O_C)|} \qquad (2.19)$$

Where *ce* is a characteristic extract and tp_{ce} is the local taxonomic precision of two concepts $c_1 \in O_C$ and $c_2 \in O_R$ as equation 2.19.

The semantic cotopy (sc) of a node is defined as the set of all its parent concepts and child concepts excluding the root and including that node’s semantic cotopy. The semantic cotopy defines the local taxonomic precision. It is defined as equation 2.20.

$$sc(c, O) \coloneqq \{c_i | c_i \in C \wedge (c_i \leq c \vee c \leq c_i)\} \qquad (2.20)$$

The taxonomic measure based on the semantic cotopy is not suitable for measuring the lexical precision and the lexical recall together. It can be applied to the common semantic cotopy (csc) for measuring the concepts and concepts hierarchies. The common semantic cotopy is as equation 2.21.

$$csc(c, O_1, O_2) := \{c_i | c_i \in C_1 \cap C_2 \wedge (c_i < 1\ c \vee c\ < 1\ c_i)\} \qquad (2.21)$$

Second, comparing concept hierarchies is used for building a global taxonomic precision measure. The taxonomic precision values are presented, which is the first building block as equation 2.22. If using the set of concepts C_C from the learned ontology, the global taxonomic precision depends on the lexical precision.

The next building block, the local taxonomic precision compares the position of a concept in the learned hierarchy and the reference hierarchy. The last building block, an estimation of a local taxonomic precision value is only used if the current concept does not exist in both ontologies.

$$TP(O_C, O_R) := \frac{1}{|C_C|} \sum_{c \in C_C} \begin{cases} tp(c, c, O_C, O_R) & if\ c \in C_R \\ max_{c' \notin C_R} tp(c, c', O_C, O_R) & if\ c \notin C_R \end{cases} \qquad (2.22)$$

Finally, creating the concrete measurement is used for completing the criteria for good evaluation measures. TP_{sc} and TR_{sc} are based on the semantic cotopy and are influenced by the lexical layer. The local taxonomic precision is computed by estimating the local taxonomic precision for all learned concepts. In the case where the result is zero, the current concept does not exist in the reference ontology. The evaluation of the lexical layer and the concept hierarchy cannot be separate. The taxonomic precision values and the common semantic cotopy are computed for the common concepts of both ontologies. TP_{sc}, TR_{sc}, TP_{csc} and TR_{csc} are defined in equation 2.23, equation 2.24, equation 2.25 and equation 2.26, respectively.

$$TP(O_C, O_R) := \frac{1}{|C_C|} \sum_{c \in C_C} \begin{cases} tp_{sc}(c, c, O_C, O_R) & if\ c \in C_R \\ 0 & if\ c \notin C_R \end{cases} \qquad (2.23)$$

$$TR_{sc}(O_C, O_R) := TP_{sc}(O_R, O_C) \qquad (2.24)$$

$$TP_{csc}(O_C, O_R) := \frac{1}{|C_C \cap C_R|} \sum_{c \in C_C \cap C_R} tp_{csc}(c, c, O_C, O_R) \qquad (2.25)$$

$$TR_{csc}(O_C, O_R) := TP_{csc}(O_R, O_C) \qquad (2.26)$$

In the Indiana Philosophy Ontology dynamic ontology (Murdock et al., 2013), the taxonomic structures are generated by machine reasoning. The expert feedbacks automatically extracted the statistical relationship from the *Stanford Encyclopedia of Philosophy*. The dynamic ontology gains the benefit of many data sources to iteratively derive the most useful domain representation. The domain experts and text corpora are the main source of data. The combination of automatic and semi-automatic methods creates a dynamic ontology.

The volatility score measures the amount of change between two or more different versions of a populated ontology. Populating an ontology means adding new instances to object assertions. The score measures the relative proportion of time *instance_of (P, Q).* The formula for assessing terms *P* and *Q* by calculating the overall volatility score is as defined in equation 2.27.

$$v(P,Q) = 1 - \frac{|x - \frac{n}{2}|}{\frac{n}{2}} \qquad (2.27)$$

Where *x* is the number of times that the *instance_of (P, Q)* is asserted. The weighting score is defined in equation 2.28.

$$v'(P,Q) = 1 - \frac{|x - \frac{m}{2}|}{\frac{n}{2}} \qquad (2.28)$$

Where *m* is the number of input sets to assert *instance_of (P, Q).* Thus the volatility measure is defined in equation 2.29.

$$volatility(z) = \frac{1}{count(P,Q)} \sum_{\forall P,Q} v'(P,Q) \qquad (2.29)$$

On the other hand, the violation score measures how well an ontology captures the semantic similarity and relationships by examining statistical measures on the corpus. For terms *S* and *G, S* is more specific than *G* in equation 2.30.

$$H(G|S) > H(S|G) \qquad (2.30)$$

The generality violation (g-violation) is measured in equation 2.31.

$$gv(S, G) = H(S|G) - H(G|S) \tag{2.31}$$

The overall g-violation is defined in equation 2.32.

$$violation_g(O) = \frac{1}{count(S,G)} \sum_{\forall S,G} gv(S, G) \tag{2.32}$$

A similarity violation (s-violation) is defined in equation 2.33.

$$sim(x_1, x_2) = \frac{2 \times log\, P(C)}{log\, P(x_1) + log\, P(x_2)} \tag{2.33}$$

Where x_1 and x_2 are entities in the taxonomy, C is the most specific class which subsumes x_1 and x_2. In order to compare an instance S to its parent G, the semantic similarity score is calculated by equation 2.34.

$$sim(S, G) = \frac{2 \times log\, P(G)}{log\, P(S) + log\, P(G)} \tag{2.34}$$

The degree of s-violation is defined in equation 2.35.

$$sv(S, G) = \frac{sim(S,G) - \mu}{\sigma} \tag{2.35}$$

Where u is the mean value of semantic distance. The semantic distance is the distance to the parent of all sibling nodes and σ is the standard deviation of this population. The final s-violation is defined in equation 2.36.

$$violation_s(O) = \frac{1}{count(S,G)} \sum_{\forall S,G} sv(S, G) \tag{2.36}$$

Experimental results were reported that the average pairwise volatility was 0.45 at the InPhO project. The decrease in s-violation means the development of denser semantic clusters subsumed under each class. The decrease in g-violation means the movement towards greater classification in the hierarchy.

CHAPTER 3 DATA-INFORMATION RETRIEVAL BASED AUTOMATED ONTOLOGY FRAMEWORK FOR SERVICE ROBOTS

Dynamic ontology is becoming more and more important in the sense of providing knowledge to support service robots. To facilitate automatic ontology to enable service robots to make decisions in dynamic environments, this research developed a Data-Information Retrieval based Automated Ontology Framework (DIRAOF). This chapter gives the overall structure of the framework.

Section 3.1 gives the background knowledge on ontology, data retrieval and information retrieval. Section 3.2 describes the overall structure of the framework. Section 3.3 introduces the components of the framework and gives an overview of each component. Section 3.4 provides the summary of Data-Information Retrieval based Automated Ontology Framework.

3.1. Background Knowledge

In information systems, ontology consists of the conceptualization of objects, instances of concepts and relations between concepts. A specific ontology represents concepts, instances and relations in a specific domain. The role of ontologies is, in humans and robot interactions, to enable the robots to understand commands from their human users and thereby to make their decisions in terms of undertaking tasks as specified by the commands (Jasper and Uschold, 1999). The components of ontology are concepts, also known as classes, collections, sets or types, instances, called objects, individuals or entities, properties, i.e. attributes, features of concepts or objects, property values, and relations between concepts and/or instances (Gómez-Pérez and Corcho, 2002; Khoo and Na, 2006). A taxonomy is a hierarchical structure of concepts in a domain with the “is-a” kind of relation between concepts.

A concept is defined as an entity of consciousness and a directly conceived or an intuited object of thought. Concepts can be physical stuff, issues, ideas, persons, processes, places, etc. and connect with the process of human cognition but they must be relevant to each other (Jakus et al., 2013). Gómez-Pérez and Corcho (2002) described a concept as any physical object existing and human beings' way of thinking in human. Concepts are shared through human language and their actions or behaviours. The explicit concept definitions can be classified into three categories (Jakus et al., 2013):

- Concepts in philosophy
- Concepts in other scientific domains and
- Concepts in knowledge representation.

Aristotle defines a concept by using genus (a kind, sort or family) and differentia (a distinguishing characteristic) (Granger, 1984). John Sowa defines concepts as a "mediator that relates symbol to its object" (Sowa, 2000). Ogden and Richards (1923) introduced the triangle that consists of object, concept and symbol. Concept explains a mediator between the symbol and its object. A concept in linguistics is a unit of meaning or conceptual meaning (Jakus et al., 2013). A concept is a model of entities from reality in the engineering field (Smith, 2004). Novak and Canas defined concepts in knowledge representation as a "perceived regularity in events or objects, or records of events or objects, designed by a label" (Novak and Canas, 2008). Normally, the label of concept is a word or a symbol. In description language, concepts (classes), roles (relations) and individuals (objects) are components of logic-based knowledge.

An ontology consists of a set of individual instances of concepts. An instance or object is a specific realization of a concept. The creation of an instance is called instantiation. Instances are collected in a knowledge base. Creating an ontology, that is creating individual instances of concepts in a specified hierarchy involves defining individual instances of concepts such as choosing a concept, creating an individual instance of that concept and filling in the property's value, and identifying and assigning relations between objects (Noy and Deborah, 2001). Creating an ontology can even involve creating concepts.

Gómez-Pérez and Corcho (2002) defined attributes as the properties of concepts. They classified attributes into four groups:

- Class attribute – attribute values attached to concepts; they are the same for all instances of a concept
- Instance attribute – different values assigned for each instance of the concept
- Same-name attributes or local attributes – attributes that share the same name but are attached to different concepts
- Global attributes – applicable to all concepts.

The Class attributes and Instance attributes are normally used in concept descriptions.

A concept cannot appear in isolation. Thus the relations between concepts are semantic relations that are meaningful associations between two or more concepts (Khoo and Na, 2006). There are two types of semantic relation between concepts and concepts (Cimiano et al., 2005). A hierarchy relation (taxonomic relations) is a relation for constructing and organising concepts into a hierarchy. It requires the discovery of the "is-a" relations. Non-hierarchy relations are the interactions between concepts which discover meronymy, attributes etc. (Wong et al., 2012). The relation between instances of concepts and concepts (an individual-to-concept relation of instantiation) is "instance-of" relations (Gangemi et al., 2001). First, the concept is selected. Next, an instance of the concept is generated by instance-of relation between instance and concept.

There are two types of OWL property (Dean et al., 2004): object properties and data type properties. Object properties usually describe the relation between two instances of a concept. For example, the "HasA" object property is the relationship between the "sponge_1" instance and the "dish_1" instance that inherited from the "sponge" concept and "dish" concept, respectively. Data type properties, however, describe relations between instance and data value. For example, the “hasProperty” data property is the relationship between the “sponge_1” instance and “green” data value.

Gómez-Pérez et al. (2004) classified ontologies into two types according to the level of detail of specifications between terms, namely lightweight ontology and heavyweight ontology. Lightweight ontologies are domain models that include taxonomic hierarchy and properties between concepts. Heavyweight ontologies attach more detail to lightweight ontologies by adding axioms and constraints to explicate terms (Gómez-Pérez et al., 2004). Ontology type can also be classified into four types based on the level of generality: top-level ontologies, domain ontologies, task ontologies and application ontologies (Auhood, 2010). Top-level ontology is a specification of a conceptualization based on linguistics independent of domain-specific concepts. Domain ontology provides a domain-specific model describing domain concepts and relations. Task ontology presents specific concepts for a task. Application ontology unites domain and task-specific ontologies

There are three types of resource for extracting knowledge when constructing ontology: structured data, semi-structured data and unstructured data (Al-Arfaj and Al-Salman, 2015). The structured data are defined as knowledge models, for example, the existing ontologies and database schema. A database is a collection of information which is managed by a database management system (Ullman and Widom, 2013). The semi-structured data are defined as mixed structured data with free text. It does not have a regular structure. Examples include Web pages, Wikipedia, and XML documents. Unstructured data are defined as textual content. They do not have a predefined data model and can appear in e-mails, notes, files, news, reports, letters, surveys, research and Web page data.

Data retrieval (DR) refers to querying and receiving data from a database. WordNet is a commonly used structured data source. It consists of English nouns, verbs, adjectives and adverbs. ConceptNet (Speer and Havasi, 2012) is another source data/knowledge resource that supports general human knowledge. It provides contextual reasoning of facts and common- sense knowledge of the real world. It is important for robots to understand the informal relations between concepts in order to perform tasks with semantic meaning. Information retrieval (IR) is applied in order to discover the semi-structured data and the unstructured data. IR is used to find material from large collections of unstructured text and semi-structured data that satisfy an information need. IR is concerned with the representation, storage, organisation and access to documents, so that it can support users in browsing or filtering document collections. Thus, the effectiveness in terms of the quality of its search results is crucial. Precision and recall are two key statistics to measure the results for a query. Precision is the fraction of the returned results that are relevant to information need. Recall is the fraction of the relevant documents in the collection that were returned by the system (Manning et al., 2008).

WordNet is selected as the first resource as concepts in WordNet are constructed by "is-a" relationships that are suitable for classifying object names from labels of physical objects. It is straightforward to create concepts and to arrange them into a concept hierarchy. WordNet also provides a semantic network with a core concept, which is called a synset. A synset is a set of one or more synonyms. Semantic relations link synsets to other synsets. However, WordNet itself is not suitable for being used whole as an ontology. This is because WordNet contains extra information apart from nouns, such as verbs, adjectives and adverbs and WordNet does not include domain-specific terminologies that are used in a robot ontology. Many object names in the household environment do not exist in WordNet. Web Documents is suitable for finding the categories of object names that are not in WordNet, as Web Documents has a huge collection of text. Web Documents is the largest electronic text source currently available to the public and a well-balanced knowledge source. The information in Web Documents is normally updated regularly.

Service robots provide services that are useful to the well-being of humans. They operate closely with humans to increase comfort or to assist the elderly or to entertain their humans. For example, Care-O-bot mobile robot (Care-O-bot, 2016) delivers food and drinks, PR2 mobile robot (PR2, 2016) navigates human environments and grasps and manipulates objects. Service robots perform tasks that are related to instances in a dynamic environment. Service robots need an ontology to support task execution and also need semantic knowledge to describe knowledge about instances in a dynamic environment where humans and robots share the same environment.

3.2 Structure of the Data-Information Retrieval based Automated Ontology Framework

The framework integrates data retrieval, information retrieval, object learning, concept creation, and relationship creation for automatically object learning ontology for service robots. It creates concepts, the concept hierarchy, relationships between these concepts and between instances of concepts. It includes an evaluation process to justify the ontology generated.

The framework subsequently consists of seven processes as shown in Figure 3.1, namely Data Input process, Automatic Ontology process, Data and Information Retrieval process, Semantic Knowledge Acquisition process, Robot Ontology process, Query process and Result Evaluation process.

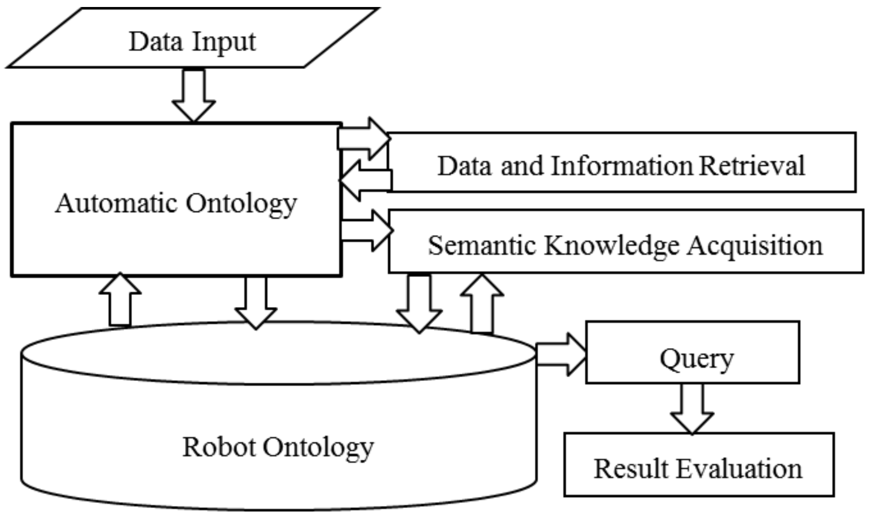


Figure 3.1 Data-Information Retrieval based Automated Ontology framework

The seven processes consist of two phases. The first phase is the automated ontology phase. It was designed as an automatic ontology creation process. All objects and properties are recognised in the Data Input process as text and are sent to the Automatic Ontology process. The Data and Information Retrieval process used WordNet for identifying object names. Object names found in WordNet are called known object names. Those that cannot be found WordNet are called unknown object names. The unknown object names are transferred to the Data and Information Retrieval process. This process returns a concept name as text to the Automatic Ontology process. The concepts and properties are created as an ontology hierarchy into the Robot Ontology using WordNet and ConceptNet. The Semantic Knowledge Acquisition represents the knowledge of objects in the form of an instance of the concept in the dynamic environment.

The second phase is the use of the automated ontology from the user point of view. It consists of three processes: Robot Ontology, Query and Result Evaluation process. First, The Query process queries the Robot Ontology, presents the query results as text to the user and sends the query results to the Result Evaluation. The Result Evaluation process retrieves the result from the Query process in order to assess the result. The following paragraphs present a broad review of Query and Result Evaluation processes. These two processes are described in full detail in Chapter 6.

The framework is a step forward to face the challenges mentioned in Chapter 1. First, the framework is able to understand and to learn objects, through the Data and Information Retrieval process without involving humans. Second, it is able to create an ontology using an innovative automatic ontology algorithm which is the core of the Automatic Ontology process. Third, it can build up semantic relations through the Semantic Knowledge Acquisition.

The framework has the following features that make it distinct from the systems reported by other researchers:

- retrieving the meaning of the random objects from Web Documents by returning a concept name,
- generating automatically ontology components by using WordNet and ConceptNet, and

- associating the semantic knowledge with instances and assigning the property values to the “HasA” and “MadeOf” property names.

The architecture and knowledge-representation framework for service robots (Rockel et al., 2013) presented how to learn from experiences and are developed in the European Union’s RACE project by applying the constraint processing technique. The main part of this architecture is the Blackboard which contains the information as ABox in description logics. The experiment-based learning consists of: a learning scenario, learning about robot activities, learning about objects and scenes and learning about environment activities. The learning about objects and scenes provides physical objects and vocabulary classification for robots. The memory-based learning method stores known instances in memory and recognises the new instances. The shape models and bag-of-feature models represent the objects. The object category knowledge is the input of the hybrid knowledge- representation and reasoning framework. Improving the robustness of the robot’s behaviour based on experience is the main focus of the RACE project. However, it is similar to the proposed framework where the Robot Ontology is represented using OWL. The RACE project does not tackle the problem of learning the unknown objects from text. It does not have the understanding feature, the creating ontology feature and the representing semantic knowledge feature.

The RoboEarth system (Tenorth et al., 2012) provides a platform for sharing knowledge about actions, objects and environment between robots. RoboEarth aims to obtain a sharable representation of the environment. It combines the experiences of many robots. It uses the semantic robot description language to describe components and the capabilities. The extension of the KNOWROB knowledge base is a representation language which is described in description logic using OWL. A KNOWROB ontology acquires knowledge from internet. It semi-automatically created the information about objects from the germandeli.com website. Thus, the RoboEarth system does not have the creating ontology feature and the representing semantic knowledge feature.

The OpenRobots Ontology Framework (ORO) (Lemaignan, 2012) comprises a common-sense ontology. The ORO common-sense ontology is designed from the OpenCyc ontology for robots. The advantage of the OpenCyc ontology is that it ensures that the knowledge can be shared or extended with well-defined semantics. As mentioned in Chapter 2, the KNOWROB ontology and ORO common-sense ontology are derived from the OpenCyc ontology and this is a predefined ontology. It can lead to situations where the robots are not able to recognise objects that do not exist in the predefined and static ontology. The ORO ontology does not have the understanding feature and the creating ontology feature.

3.3 Processes in the Data-Information Retrieval based Automated Ontology Framework

The following sub-sections discuss the seven processes, namely Data Input process, Automatic Ontology process, Data and Information Retrieval process, Semantic Knowledge Acquisition process, Robot Ontology process, Query process and Result Evaluation process.

3.3.1 Data Input Process

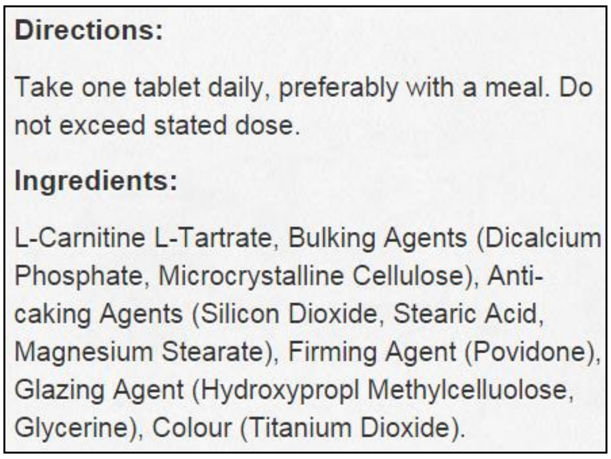
Generating an ontology using information from text requires object names and property values of physical objects. However, when robots approach objects, they often see the labels of objects. In some cases, the labels contain object names and/or property values together with much more information, and in many cases they do not give object name and/or property value. Figure 3.2 (a) shows a label that has the object name and property values. Figure 3.2 (b) shows a label that has object names but without any property value. Figure 3.2 (c) shows a label that has property values but without an object name. Figure 3.2 (d) shows a label that does not have an object name and property value. Therefore, there is a need for the robots to discover object names and property values from labels. The inputs of this process are labels and the outputs are an object name and property values.

Labels can also contain other information in addition to object names and/or property values if they are not on the labels. To understand a given object, robots need to extract only the information that is related to object name and property values from its label. In this study, an assumption of noun relating to object name and adjectives relating to properties is made.

When entering a room, a household robot normally makes a scan to recognise objects and furniture in the room (Ji et al., 2012). The scan returns the labels and the locations of all objects in the room. The robot then stores all label and location information collected in a text file. The file contains three pieces of information, namely, the date and time of the scan process, and the location and label text. Figure 3.3 shows an example text file. The first line "20150819113959" is the date and time information, meaning that the scan process took place on 2015, August 19 at 11:39:59". Then, each line of text represents the location and the label text of a physical object. For example, on the second line, "1, 0, 1" represents the location of a physical object, and "Cute Press EVORY BB powder SPF 25" is the label text of the object. The last line of Figure 3.3 shows that no label text was found for a particular object in the scan.

(a) Cute Press EVORY BB powder SPF 25

(b) Tucher Helles Hefe Weizen



Directions:

Take one tablet daily, preferably with a meal. Do not exceed stated dose.

Ingredients:

L-Carnitine L-Tartrate, Bulking Agents (Dicalcium Phosphate, Microcrystalline Cellulose), Anti-caking Agents (Silicon Dioxide, Stearic Acid, Magnesium Stearate), Firming Agent (Povidone), Glazing Agent (Hydroxypropl Methylcelluolose, Glycerine), Colour (Titanium Dioxide).

(c) Directions: Take one tablet daily, preferably with a meal. Do not exceed stated dose. Ingredients: L-Carnitine L-Tartrate, Bulking Agents (Dicalcium Phosphate, Microcrystalline Cellulose), Anti-caking Agents (Silicon Dioxide, Stearic Acid, Magnesium Stearate), Firming Agent (Povidone), Glazing Agent (Hydroxypropl Methylcelluolose, Glycerine), Colour (Titanium Dioxide).

(d) no label

Figure 3.2 Example of labels of physical object

```
20150819113959
1, 0, 1 Cute Press EVORY BB powder SPF 25.
1, 1, -1 Tucher Helles Hefe Weizen.
1, 1, 1 Directions: Take one tablet daily, preferably with a meal. Do not exceed
stated dose. Ingredients: L-Carnitine L-Tartrate, Bulking Agents (Dicalcium
Phosphate, Microcrystalline Cellulose), Anti-caking Agents (Silicon Dioxide,
Stearic Acid, Magnesium Stearate), Firming Agent (Povidone), Glazing Agent
(Hydroxypropl Methylcelluolose, Glycerine), Colour (Titanium Dioxide).
1, 1, 0 None.
```

Figure 3.3 Example of text file

The Data Input process consists of two following sub-processes. In the first sub-process, POS tagging (Han, 2009) is applied to selected cardinal numbers, adjectives and nouns for each line of the text file. This work focuses on cardinal numbers (CD), adjectives (JJ), singular or mass nouns (NN), plural nouns (NNS), singular proper nouns (NNP), plural and proper nouns (NNPS) based on Brown and Penn Treebank tags (Taylor, 2003).

There are two types of cardinal number that are used to point out properties of an object: time of scan process and location of object. Though the time information will not be used to recognise object names and properties, it will be used to add semantic relation between objects. It can be detected from the first line of the text file. The cardinal number of time is going to be a data value of the "hasTime" property of an object. The location information, on the other hand, is going to be a data value of the "hasLocation" property of an object. It can be detected from the beginning of each line, which gives the value of the x, y, z coordinates of an object in a global reference system defined for the work space of a robot (Martinez and Fernandez, 2013).

Adjectives give properties of physical objects. All property names are sent to the Automatic Ontology process where the property names are classified and assigned as data values to five predefined data type properties: "hasTime", "hasLocation", "hasColour", "hasShape" and "hasProperty". Datatype properties are selected following the rule-based approach. These data type properties relate to the Semantic Knowledge Acquisition. Due to the property names are value of data type property that is selected from the predefined data type properties in the Property Creation.

Noun indicates a physical object name. In the simple case where one noun is found in a line, the noun is considered as an object name. In the case where more than one noun is found, the group nouns are sent to Data and Information Retrieval process for further investigation in order to find a category name.

Figure 3.4 presents the result after applying the POS tagging to the instance that is given in Figure 3.3. As mentioned earlier, the first line is date and time and the beginning of each line is location of physical object. The underlined texts are adjectives and the bold texts are nouns. Figure 3.4 shows verb (VB), adverb (RB), preposition (IN), determiner (DT), interjection (UH) and conjunction (CC) based on Brown and Penn Treebank tags (Taylor, 2003).

20150819113959/CD

1/CD, /, 0/CD, /, 1/CD Cute/JJ **Press/NNP EVORY/NNP BB/NNP powder/NN** SPF/, 25/CD

1/CD, /, 1/CD, /, -1/CD **Tucher/NN Helles/NNS Hefe/NN Weizen/NN**

1/CD, /, 1/CD, /, 1/CD **Directions/NNS**: /: Take/VB one/CD **tablet/NN** daily/RB, /, preferably/RB with/IN a/DT **meal/NN**. /. Do/VB not/RB exceed/VB stated/VB **dose/NN**./. **Ingredients/NNS**:/: L-Carnitine/ L-Tartrate/., /, Bulking/UH **Agents/NNS** (/ (Dicalcium**/NN Phosphate/NNP**, /, **Microcrystalline/NNP Cellulose/NN**)/), /, Anti-caking/CC Agents/NNPS (/ (Silicon**/NNP Dioxide/NN**, /, Stearic/JJ **Acid/NN**, /, **Magnesium/NN** Stearate/IN)/), /, Firming/VB **Agent/NNP** (/ (Povidone**/NNP**)/), /, Glazing/VB **Agent/NNP** (/ (Hydroxypropl**/NNP Methylcelluolose/NNP**, /, **Glycerine/NN**)/), /, Colour/JJ (/ (Titanium**/NNP Dioxide/NN**)/)./.

1/CD, /, 1/CD, /, 0.25/CD

Figure 3.4 Results using Brown and Penn Treebank tags

The segmentation and repetition is the second sub-process. It aims to select the date and time, location, and property values, object name and to delete the duplicated words and the lines that do not have nouns and adjectives. The resultant lines, as shown in Figure 3.5, are then sent to the Automatic Ontology process.

```
20150819113959
1, 0, 1 cute Press EVORY BB powder
1, 1, -1 Tucher Helles Hefe Weizen
1,1,1 Directions tablet meal dose Ingredients Agents Dicalcium Phosphate
Microcrystalline Cellulose Silicon Dioxide Stearic Acid Magnesium Agent
Povidone Agent Hydroxypropl Methylcelluolose Glycerine Colour Titanium
Dioxide
```

Figure 3.5 Text file sends to the Automatic Ontology process

3.3.2 Automatic Ontology Process

The components of an ontology normally include concepts, relations between concepts, instances, properties and property values. The Automatic Ontology aims at automatically constructing the components of an ontology from the object names and property values given by the Data Input process. In the framework developed in this research, this process is implemented in the following five modules: (1) Concept Creation module, (2) Relation Creation module, (3) OWL Creation module, (4) Instance Creation module and (5) Property Creation module. Information exchanges between the modules and between the Automatic Ontology process and other processes are shown in Figure 3.6.

To create concepts and the relationships between concepts, it is necessary to understand the meanings of object names. This involves Data and Information Retrieval because an object name often does not indicate object types. Concept Creation sends the object name to the Data and Information Retrieval process to seek more information relevant to the meaning of the object names. The latter process searches for the relevant information and works out a concept name that indicates the types of the objects. It also identifies the “is-a” relation between concepts. The process returns concepts to the Concept Creation module and “is-a” relation to the Relation Creation module.

The Concept Creation module and Relation Creation module then work together to construct the concept hierarchy based on the concepts and the "is-a" relations. Relation Creation creates "HasA" relation and "MadeOf" relation between the concept and the relevant concept from ConceptNet. If the relevant concept can be found in the Robot Ontology, the "HasA" relation and/or "MadeOf" relation is created between the concept and the relevant concept.

The OWL Creation module represents concepts, relations obtained from the Relation Creation module with OWL tags. The latter can then be sent and stored as part of the Robot Ontology.

The Instance Creation module receives the concept name either from Concept Creation if the concept is a new one or from the Robot Ontology process in the case where it is an existing one. It names the object for which a concept is created as the instance of that concept. For example, given an object of mouse, the concept of "mouse" is created and that particular mouse, instance of the "mouse" concept, is named "mouse_1".

Property values from Data Input are handled in the Property Creation module which identifies a property name for the instances. Property values are classified into three predefined property names. Property Creation assigns property values to property names and concepts to OWL tags. OWL tags are sent to Semantic Knowledge Acquisition process.

The following is an example showing the operation of the Automatic Ontology process. After the Automatic Ontology process receives the text file from the Data Input process, as shown in Figure 3.5, the object name, "Tucher Helles Hefe Weizen" is sent to Data and Information Retrieval in order to obtain the meaning of the object name. The Data and Information Retrieval process returns the "beer" concept to the Concept Creation process. Before creating the concept, Concept Creation sends the "beer" concept to check its existence in the Robot Ontology. If the "beer" concept is a new concept, it cannot be found in the Robot Ontology, and then the "beer" concept is created. The Instance Creation module receives the "beer" concept from the Concept Creation process in the case that it is an existing one. It names the object for which the “beer” concept is created as the instance of that concept. Thus, the concept of beer is created and that particular beer is named “beer_1” and instance name is sent to Property Creation.

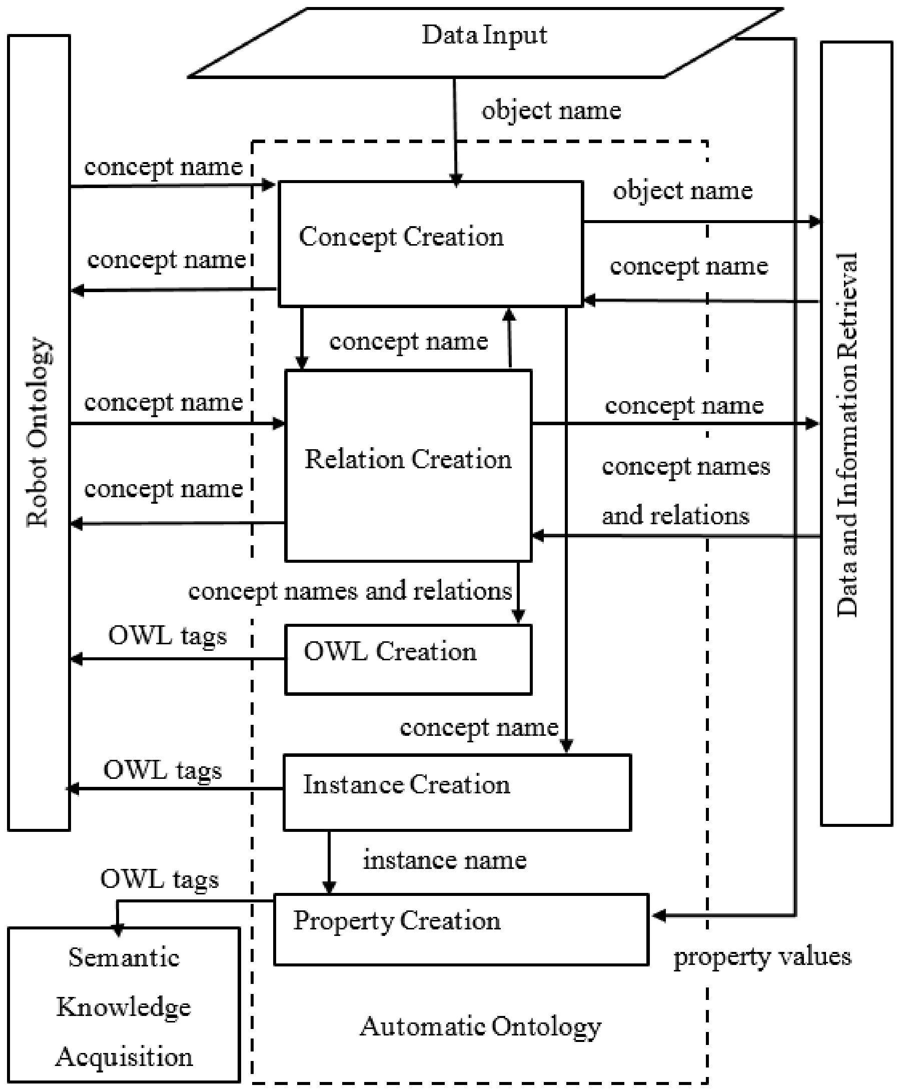


Figure 3.6 The Automatic Ontology process

The “beer” concept is sent to Relation Creation in order to create relations between the “beer” concept and other concepts. The Data and Information Retrieval supports the Relation Creation for creating the hierarchy and non-hierarchy relations.

For the hierarchy construction, the Data and Information Retrieval returns the "alcohol", "drug" and "fluid" concepts and the "is-a" relations between "beer" and "alcohol", "drug" and "fluid" concepts. Relation Creation sends all concepts to the Robot Ontology. If the concept cannot be found in the Robot Ontology, then Concept Creation creates a new concept and creates an "is-a" relation. Otherwise, Relation Creation creates an "is-a" relation between the newly created concept and the existing concept.

Data and Information Retrieval returns the "HasA" relation with the "water" and "alcohol" concepts for the non-hierarchy construction. Relation Creation sends the "water" and "alcohol" concepts to the Robot Ontology. The "alcohol" concept exists in the Robot Ontology, and then Relation Creation creates "HasA" relation between the "beer" and "alcohol" concepts. Concept Creation does not create the relation between concepts, if the "water" concept does not exist in the Robot Ontology.

The OWL Creation module receives all concepts and relations from Concept Creation and Relation Creation and converts concepts and relations between concepts into OWL tags. These tags are sent and stored as part of the robot ontology.

Property Creation receives the property values from Data Input. The Property Creation module identifies "20150819113959", "1, 0, 1" and "Cute" as property values of the "hasTime", "hasLocation" and "hasProperty" property names, respectively. The property name and property value are added into the "beer_1" instance and the semantic knowledge is updated in the Semantic Knowledge Acquisition process.

3.3.3 Data and Information Retrieval Process

As mentioned in Section 3.3.2, given an object name, the Data and Information Retrieval process looks for extra information, and recognises concepts, and hierarchy and non-hierarchy relations. It returns the concepts and the relations back to Automatic Ontology to generate the ontology components.

Data and Information Retrieval acquires data or information from the structured, semi-structured and unstructured data sources. WordNet, ConceptNet and Web Documents are used as the data and information sources in this framework. WordNet and ConceptNet contain structured data, while Web Documents are regarded as either semi-structured or unstructured data.

Figure 3.7 shows the Data and Information Retrieval process. It starts with the Web Documents search to obtain a category name in order to assign it to a concept name. It then employs WordNet for finding out concept names and hierarchy relations and ConceptNet for non-hierarchy relations.

With the obtained Web Documents, the Data and Information Retrieval process first searches with the keyphrase: object name “is”. The key noun after “is” in the obtained documents is selected as a candidate category name. Then the frequency of occurrence of each candidate category name is counted. The one with the highest frequency is selected as the category name.

Category names which are obtained from Web Documents are sent into WordNet in order to find the parent of category name for a particular physical object. In the case where the parent of the category name can be found in WordNet, Concept Creation assigns the category name as the concept name. There are five commonly used semantic relations for nouns: synonym, hyponym, hypernym, holonym and meronym. A synonym is a word or phrase that means exactly or nearly the same as another word or phrase. A hypernym is a word with a broad meaning. A hyponym is a word of more specific meaning than a hypernym. A holonym is used for naming the whole of which a given word is a part. A meronym is used for naming a part of a larger whole (Miller, 1995). A hypernym provides the broad meaning of a concept. Thus concepts and taxonomic relations in WordNet are retrieved by using hypernyms for constructing new concepts and relations between the new concepts into a concept hierarchy.

After that, the concept name is searched in WordNet again to retrieve the hypernyms. The retrieved hypernyms, concept names, are physical entities with “is-a” relations. Then the concept names and “is-a” relations are sent to Relation Creation in Automatic Ontology in order to create the concept hierarchy.

ConceptNet (Speer et al., 2012) is the large-scale common- sense knowledge bases that support general human knowledge. It provides contextual reasoning of facts and common- sense knowledge of the real world. It is important for a robot to understand the informal relations between concepts in order to perform tasks with semantic meaning. The free-text relation in ConceptNet is defined into 21 relations. Each node in ConceptNet uses an English fragment which consists of four syntactic constructions: noun phrases, verbs, prepositional phrases and adjectival phrases. The idea of using ConceptNet is to create the semantic meaning between two physical objects with the selected relations.

For non-hierarchy relation purposes, a concept name is searched for with “HasA” and “MadeOf” relations in ConceptNet to retrieve the relevant concept names. After that, the relevant concept names are searched for in Robot Ontology. The relevant concept names that can be found in Robot Ontology are created with “HasA” or/and “MadeOf” relations in Relation Creation in Automatic Ontology.

Information exchange between the modules and between the Data and Information Retrieval processes and the Automatic Ontology process are shown in Figure 3.7.

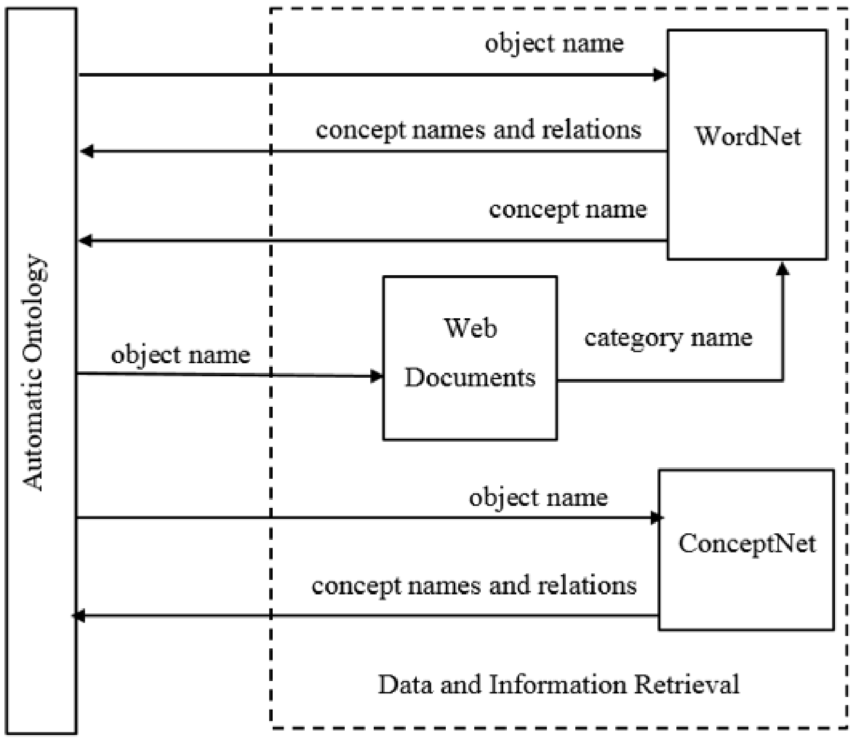


Figure 3.7 The Data and Information Retrieval process

For example, giving “Tucher Helles Hefe Weizen” to the Web Document search within Data and Information Retrieval returns the category name of “beer” as the concept name and sends this concept name to WordNet. WordNet provides “alcohol”, “drug” and “fluid” as parent concepts. Therefore, the “beer” category name can be assigned as the concept name. The “beer” concept will be created as a concept. Next, ConceptNet produces the non-hierarchy relations between the concept of “beer” and the existing concepts in the Robot Ontology with “HasA” relation and “MadeOf” relation. Suppose the “water” and “malt” concepts are two relevant concept names. ConceptNet has “beer “HasA” water” and “beer “MadeOf” malt”. The non-hierarchy relations “beer “HasA” water” and “beer “MadeOf” malt” can be confirmed and sent back to Automatic Ontology.

3.3.4 Semantic Knowledge Acquisition Process

Semantic knowledge represents the properties of physical objects in the surrounding environment. Obtaining the semantic knowledge of physical objects requires concept names, property names and property values, as the inputs to Semantic Knowledge Acquisition. The output of this process is a set of OWL tags that represent the instances of concepts for each physical object with property names and property values in the scene, such as

```
“<ClassAssertion>
        <Class IRI="#powder"/>
        <NamedIndividual IRI="#powder_1"/>
</ClassAssertion>

<DataPropertyAssertion>
        <DataProperty IRI="#hasTime"/>
        <NamedIndividual IRI="#powder_1"/>
        <Literal datatypeIRI="&rdf; PlainLiteral">20150819113959</Literal>
</DataPropertyAssertion>” tags.
```

The Semantic Knowledge Acquisition consists of four modules as shown in Figure 3.8. Firstly, the Ontology Updating module receives semantic knowledge from Automatic Ontology and associates semantic knowledge with Robot Ontology. Semantic Representation receives the concept name from Query and receives tags from Robot Ontology. The NFile module compares the instances of concepts from time to time. Finally, the Object Prediction module predicts the current location of the instance of the concept.

For example, the “powder” concept is generated by Automatic Ontology. Instance Creation sends the “powder_1” instance to Semantic Knowledge Acquisition. Property Creation sends the “hasTime”, “hasLocation” and “hasProperty” properties with “20150819113959”, “1, 0, 1” and “cute” values, respectively, to Semantic Knowledge Acquisition. This information will be represented in OWL tags and sent to Robot Ontology.

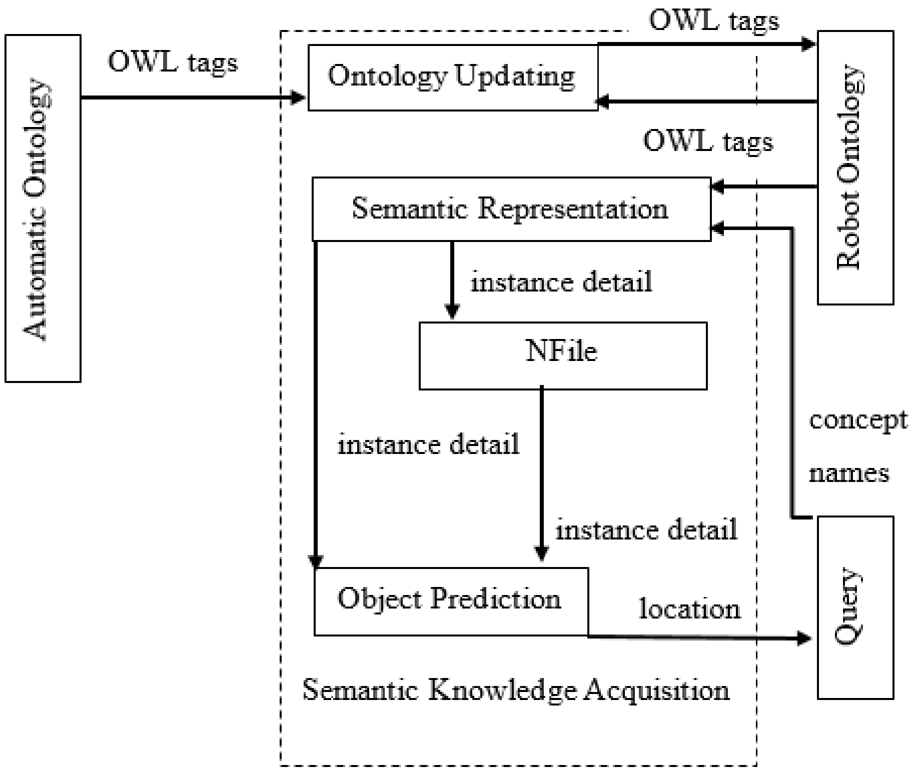


Figure 3.8 The Semantic Knowledge Acquisition process

3.3.5 Robot Ontology Process

Robot Ontology is designed with two levels, namely, the Ontology Level and the Semantic Knowledge Level, in order to support service robots to competently perform everyday tasks. The Ontology Level presents a concept hierarchy, concepts and relations between concepts. The contents of this level are created through the Automatic Ontology process. The Semantic Knowledge Level presents instances of concepts and their properties. The contents of this level are created through the Semantic Knowledge Acquisition process. Figure 3.9 shows an example. At the Ontology level, the “powder” concept is created with “is-a” relations with the “matter” concept and “drug” concepts. The “matter” concept has an “is-a” relation with the “physical entity” concept and the “drug” concept has an “is-a” relation with the “physical entity” concept. The concept hierarchy is organised from the “powder”, “matter”, “drug” and “physical entity” concepts with “is-a” relations as described earlier. At the Semantic Knowledge Level, the “powder_1” instance is an instance of the “powder” concept. The “powder_1” instance has "hasTime" data type property with "20150819113959" data value, "hasLocation" data type property with "1, 0, 1" data value and "hasProperty" data type property with "cute" data value.

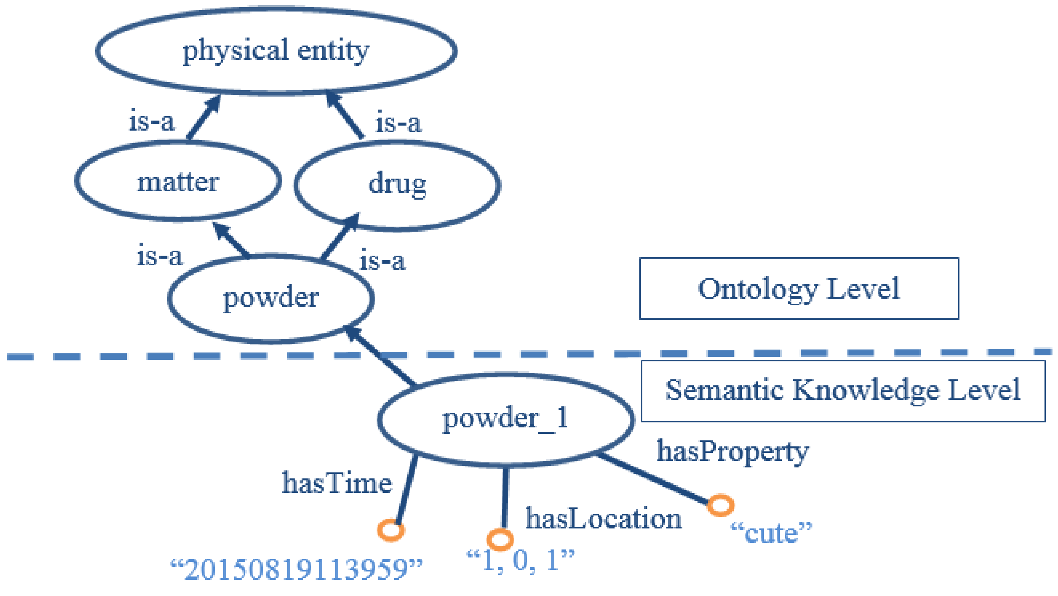


Figure 3.9 The Robot Ontology structure

The Robot Ontology process aims to store the newly created robot ontology contents according to the two levels. The key in this process is a management component that keeps the relevant information and eliminates redundancy. For the purpose of redundancy elimination, the ontology management receives concept names from Concept Creation in order to check the existence of the concept. If there is an existing concept name in Robot Ontology, then the concept name is not created in Concept Creation but an instance of the concept is created in Instance Creation. This component also manages queries that are raised due to the use of the ontology and because of evaluation of the created ontology contents. The ontology management obtains the queries from Query and searches for the object name in the Robot Ontology Level to retrieve the concept names and their synonyms' concept names and the Semantic Knowledge Level to retrieve the instances of the concepts names, data type properties and values. First, the concept name is selected by key word matching with the object name. Next, the synonymous concept names of this concept are found by sending via Automatic

Ontology through Data and Information Retrieval. After that, the “instance-of” relation between the concept and the instance is found to retrieve the instances of the concept name and the instance of synonymous concept names. The information exchange between the management and other processes can be seen in Figure 3.10. Then Robot Ontology returns the results to other processes.

For example, the ontology management obtains the “powder” concept name from Automatic Ontology. It has “20151130160000” and “1, -1, 1” as value data of “hasTime” and “hasLocation” data type properties, respectively. The “powder” concept already exists in Robot Ontology so the ontology management sends the “powder” concept to Instance Creation to create the “powder_2” instance and “20150819113959” and “1, 0, 1” as value data of “hasTime” and “hasLocation” are created in Semantic Knowledge Acquisition.

The “powder” object name is searched for in Robot Ontology and the “powder” concept is returned. Next, the synonyms of the “powder” concept are found in Data and Information Retrieval via Automatic Ontology.

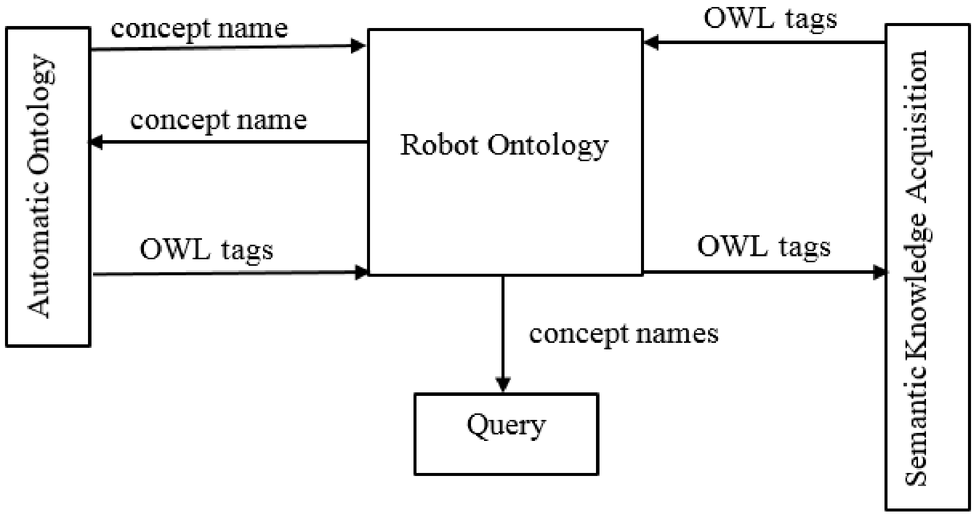


Figure 3.10 The Robot Ontology process

3.3.6 Query Process

This process serves two purposes. First, the newly created ontology contents in terms of concepts, instances, relations, etc. can be justified through raising automatic queries to this process. Second, Query can inquire in order to search for child concepts in the ontology. The inputs of this process are concept names from Robot Ontology. The output will be the set of the concept and its child concepts. The Query process mainly performs retrieving and searching results. Then having received the concept name, as shown in Figure 3.11, there are two tasks: the retrieving task and the searching task.

For the retrieving task, Query retrieves the concept name from the Ontology Level of Robot Ontology to send the concept name to Semantic Knowledge Acquisition. The Ontology Level links to instances of concepts in the Semantic Knowledge Level. The instances of the concepts are selected in order to calculate the predicted location of the instance. Query also receives the predicted location of the instance back from Semantic Knowledge Acquisition.

For the searching task, the concept name is passed into Robot Ontology in order to search for child concepts. Query sends the set of the concept and its child concepts to Result Evaluation (for details refer to Chapter 5).

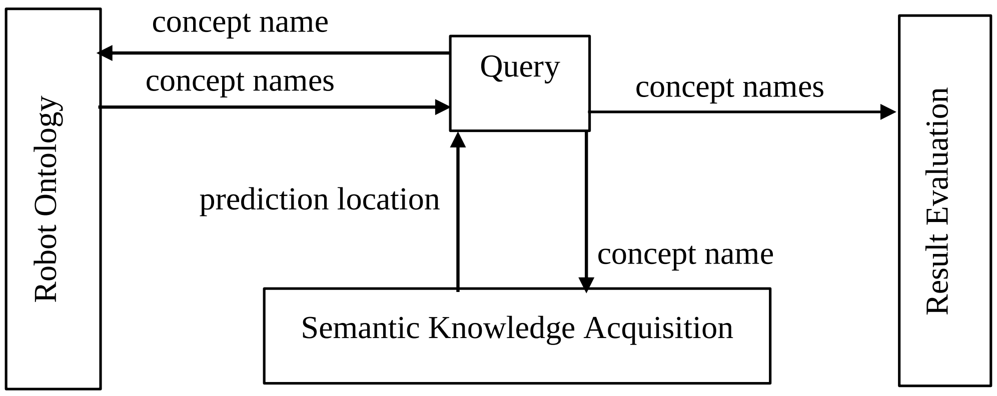


Figure 3.11 The Query process and Result Evaluation process

3.3.7 Result Evaluation Process

Before the ontology becomes available to or is integrated into other applications, it has to be evaluated during its development lifecycle (Fahad and Qadir, 2008). Result Evaluation aims at evaluating structural aspects of the ontology. Dellschaft and Steffen (2008) classified evaluation methods into two main categories: the quality assurance during the ontology engineering process (task-based, corpus-based and criteria-based) and the comparing ontology learning (domain expert or gold standard-based).

Result Evaluation is the process to verify the correct creating of the content of Robot Ontology. In order to justify the dynamic ontology, concept names and their child concepts are required from the robot itself through the Query process.

3.4 Summary

The proposed framework of dynamic ontology is presented in order to gain a better understanding of objects, to create an automatic ontology and to represent semantic knowledge for a robot. The studies on the data and information retrieval show that the retrieved concept name is required in order to understand the physical object and to build the concept into a concept hierarchy. The semantic knowledge represents the instances of concepts in the environment to support the tasks for robots. Therefore, a new framework of Data-Information Retrieval based Automated Ontology for Service Robots has been proposed in this study.

The Automatic Ontology process and the Data and Information Retrieval process are investigated in Chapter 4 as Concept Creation. The Semantic Knowledge Acquisition is described in Chapter 5. The Query and the Result Evaluation processes are presented in Chapter 6.

CHAPTER 4 CONCEPT CREATION

The Automatic Ontology process and the Data and Information Retrieval process within the DIRAOF framework are designed to create concepts and components of the ontology. This chapter details the two processes and how they work together on concept creation.

4.1 Background Knowledge

WordNet (Miller, 1995) is a lexical database. There are four syntactic categories, namely English nouns, verbs, adjectives and adverbs. They are organised into synsets (sets of synonyms). A synset is a set of one or more synonyms. Semantic relations link synsets to other synsets. There are six semantic relations: synonymy, antonym, hyponymy, meronymy, troponym and entailment. Synonymy is the relation between words which represent the same concept. Antonym is relation between words which have the opposite meaning. Hyponymy and hypernym are super–subordinate relations which arrange the meanings of nouns into a hierarchical structure. Meronymy and holonym are part–whole relations. The meronym represents the part name and the holonym represents the whole name. Troponym is the relation between a verb of a more precise manner and a verb of a more generalised meaning, such as whisper and talk. Entailment is a relation between verbs, such as sleep and snore.

The WordNet web application (http://wordnetweb.princeton.edu/perl/webwn) provides the meaning words and concepts via a browser. It allows querying the WordNet lexical database via a graphical interface. The WordNet application on Windows provides a user interface and can be installed on a personal computer for searching for the meaning of words and concepts. The WordNet database can be downloaded and installed in order to utilise it for developing programming for a specific purpose.

The WordNet database is an ASCII-format database. WordNet provides a C application program interface for accessing the WordNet database. There are many functions that can be used by a developer. For example, findtheinfo () function is the primary search algorithm for using with database interface applications.

ConceptNet (Speer and Havasi, 2012) is a large-scale common- sense knowledge base. It provides contextual reasoning of facts and common- sense knowledge of the real world. The collection of simple facts about people and everyday life can be called common- sense knowledge. ConceptNet is collected from many sources of knowledge: sister projects to MIT's Open Mind Common Sense (OMCS), WordNet3.0, DBPedia and Wikipedia's free text.

The structure of ConceptNet is a network of labelled nodes and edges. The nodes (concepts) represent words, word senses, and short phrases. The edges are pieces of common- sense knowledge. An edge connects concepts to each other with a relation. Example of the standard relations in ConceptNet are: "IsA", "UsedFor", "HasA", "CapableOf", "PartOf", "MadeOf". An assertion is a sentence that is expressed by a relation between two concepts such as the "IsA" relation presents "beer is a kind of beverage" where the "/r/IsA" relation connects "/c/en/beer" to "/c/en/beverage". Every object in ConceptNet has a URI. It is structured like a path that provides a standard place to look it up. For example, "/c/en/beer" is the URI of the "beer" concept in English (ConceptNet 5, 2015).

The ConceptNet web application (http://conceptnet5.media.mit.edu/) provides the user interface for searching for a concept in ConceptNet. ConceptNet Web API queries knowledge about any concept in ConceptNet. There are three methods: lookup, search and association – for accessing data through the ConceptNet Web API. The current development of ConceptNet (ConceptNet 5, 2015) is an open-source project that is available on https://github.com/commonsense/conceptnet5.

Search method accesses data through the ConceptNet 5 Web API. The base URL for searching is http://conceptnet5.media.mit.edu/data/5.4/search. The arguments specify what to search for. The URI argument can be "id", "uri", "rel", "start", “end”, “dataset” and “license”. The “limit = n” argument is the number of results. The “offset=n” argument indicates the first n results that are skipped. For example, the “rel” argument is the relation name “IsA”. The “start” argument means that the “beer” concept has “IsA” relation with the concept in ConceptNet. The “limit” argument is set to 10 results. The ConceptNet 5 Web API URL is http://conceptnet5.media.mit.edu/data/5.4/search?rel=/r/IsA&start=/c/en/beer&limit=10.

A Web Document is a collection of text, images, audio video, hyperlinks, etc. that is created from web programming languages. A website is a collection of Web Documents (web pages) that are interconnected by hyperlinks. The Uniform Resource Locator (URL) indicates the unique website. The internet is a collection of computers or networking devices that connect websites together. The internet allows people to share information through websites. Websites contains Web Documents that are semi-structured or unstructured data. A semi-structured data presents data and schema together. Examples of semi-structured data are WordNet, HTML and XML documents. An unstructured data is text documents and web page data. Information retrieval is needed in order to retrieve data from Web Documents. Information retrieval focuses on retrieving documents based on the content of their semi-structured or unstructured components.

A search engine, such as Google, Yahoo Search, Bing, Ask etc., searches for websites by keywords. The search results show the relevant websites that contain the keywords. The contents from the websites are extracted by using a keyphrase. The keyphrase is a set of phrases that indicate the requisite data from the search results. The keyphrase extraction processes two steps: selecting candidate words and phrases and determining the candidate keyphrase. The first step, selecting candidate words and phrases, involves removing stop words and indicating the POS tagging process. Determining the candidate keyphrase is corrected by using supervised or unsupervised methods.

The ontology is represented with OWL 2 Web Ontology Language (OWL 2) using Web Ontology Language and eXtensible Markup Language (XML) as

OWL/ XML syntax. OWL 2 is an ontology language with formally defined meaning that provides concepts (classes), properties, instances (individuals) and data values. OWL 2 associates with an ontology document. An ontology document contains the physical text documents. The text documents are written in the OWL 2 syntax (OWL 2, 2016). The elements in the ontology document can be assigned semantic meaning. This can be used to check concept consistency. Concept consistency is no contradiction among the definitions of concepts.

OWL/XML syntax applies the structural specification of OWL 2 and an XML schema. An XML schema describes the structure of an XML document. The XML schema language is created as XML SchemaDefinition (XSD). There are six types of entity: Class, Datatype, ObjectProperty, DataProperty, AnnotationProperty and NamedIndividual as given in the XML SchemaDefinition:

```
<xsd:group name="Entity">
   <xsd:choice>
     <xsd:element ref="owl:Class"/>
     <xsd:element ref="owl:Datatype"/>
     <xsd:element ref="owl:ObjectProperty"/>
     <xsd:element ref="owl:DataProperty"/>
     <xsd:element ref="owl:AnnotationProperty"/>
     <xsd:element ref="owl:NamedIndividual"/>
   </xsd:choice>
 </xsd:group>.
```

An OWL tag is used to represent a concept as:

```
<Declaration>
       <Class IRI="#concept name"/>
</Declaration>.
```

The Class element is a sub-element of the Declaration element. It has an IRI attribute that contains concept name. Internationalized Resource Identifiers (IRI) is an attribute of concept. IRI extends Uniform Resource Identifiers (URIs) by using the Universal Character Set. URIs use ASCII, which is limited, but IRIs use the Universal Character Set, which is Unicode/ISO character. IRIs is represented by the IRI UML class. The string value of the IRI indicates the same structure if two string values of IRIs are the same.

An OWL tag is used to represent a hierarchy relation as:

```
<SubClassOf>
        <Class IRI="#child concept"/>
        <Class IRI="#parent concept"/>
</SubClassOf>.
```

The SubClassOf element has a Class element as a subclass. The first Class element has an IRI attribute that contains child concept. The second Class element has an IRI attribute that contains parent concept.

An OWL tag is used to represent an individual (instance) as:

```
<Declaration>
        <NamedIndividual IRI="#individual name"/>
</Declaration>
<ClassAssertion>
        <Class IRI="#concept name"/>
        <NamedIndividual IRI="#individual name"/>
</ClassAssertion>
```

The NamedIndividual element is a sub-element of the Declaration element. It has an IRI attribute that contains individual name. The ClassAssertion element has Class and NamedIndividual elements. The Class element has an IRI attribute that contains concept name. The NamedIndividual element has an IRI attribute that contains individual name. There are two types of property: object property and datatype property. An Object property presents the relation between two classes. An OWL tag is used to represent an object property as:

```
<Declaration>
        <ObjectProperty IRI="#relation name"/>
</Declaration>
<EquivalentClasses>
        <Class IRI="#concept name1"/>
            <ObjectSomeValuesFrom>
                    <ObjectProperty IRI="#relation name"/>
                    <Class IRI="#concept name2"/>
            </ObjectSomeValuesFrom>
</EquivalentClasses>.
```

The ObjectProperty element is a sub-element of the Declaration element. It has an IRI attribute that contains relation name. Concept name1 has a relation name with concept name2. The Class element is a sub-element of the EquivalentClasses element. It has an IRI attribute that contains concept name1. The ObjectSomeValuesFrom element is a sub-element of the Class element. It has two sub-elements: ObjectProperty and Class elements. The ObjectProperty element has an IRI attribute that contains relation name. The Class element has an IRI attribute that contains concept name2. Datatype property presents relation between data property of individual and their values. The DataPropertyAssertion element has three elements: DataProperty, NamedIndividual and Literal. An OWL tag is used to represent a data property as:

```
<DataPropertyAssertion>
        <DataProperty IRI="# relation name "/>
        <NamedIndividual IRI="#individual name"/>
        <Literal datatypeIRI="&rdf;PlainLiteral">property value</Literal>
</DataPropertyAssertion>.
```

The DataProperty element has an IRI attribute that contains relation name. The NamedIndividual element has an IRI attribute that contains individual name. The Literal element has a datatypeIRI attribute that contains the &rdf;PlainLiteral value. The Literal element has a value that contains property value.

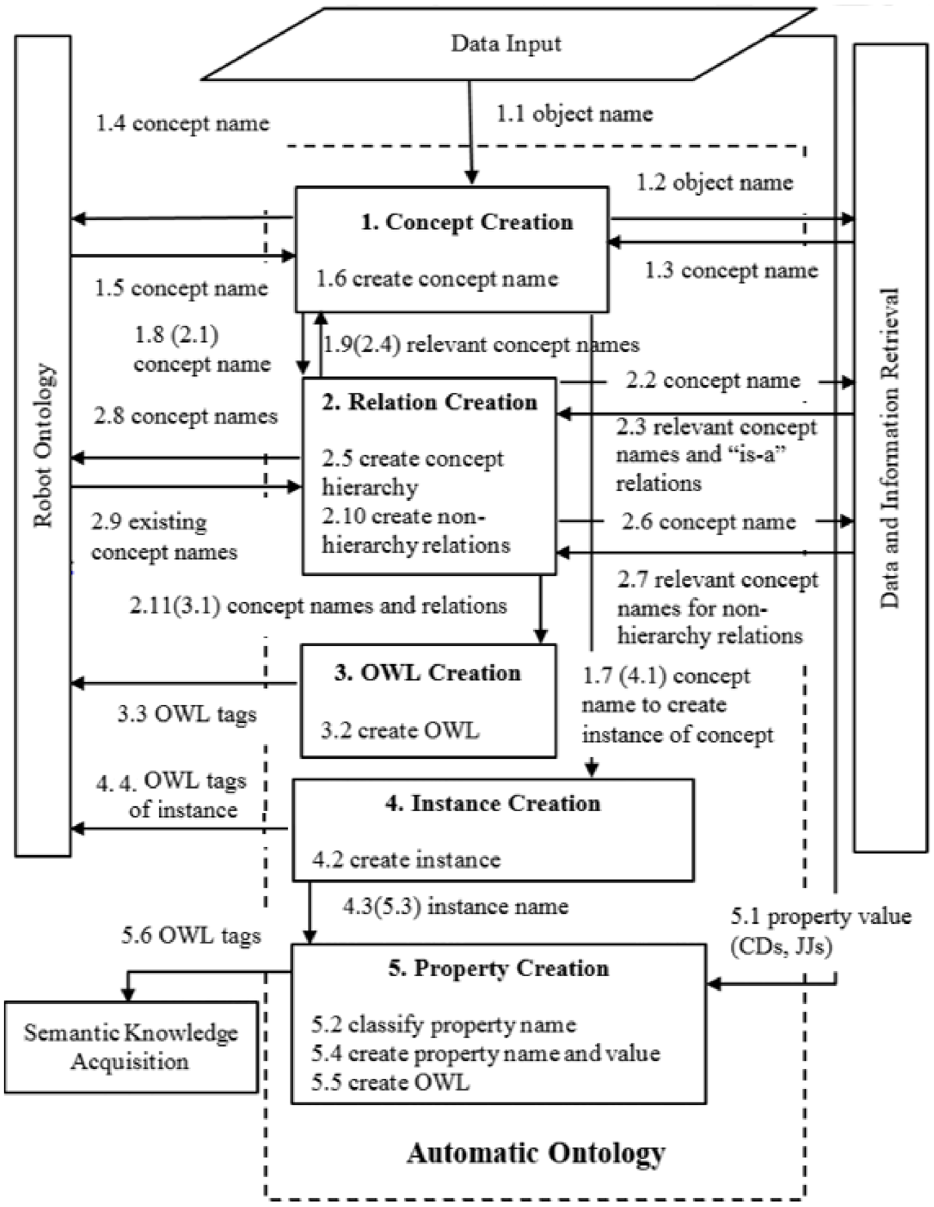
4.2 Automatic Ontology Process

Concepts and components of an ontology such as relations between concepts, instances and properties of instances are obtained based on information found in text labels of objects in the Automatic Ontology process of the framework. Text labels can give and, in most case, imply an object’s name and properties, which are closely related to concepts and components of an ontology. The Automatic Ontology process identifies concepts and components of the ontology by running data and information retrieval from WordNet, ConceptNet and Web Documents based on the information on text labels. Images may also imply concepts. However, different objects that share similar images can lead to the same concept. Moreover, images cannot give the invisible properties of an object.

The Automatic Ontology aims at automatically constructing concepts, instances, properties, property values and relations between concepts from the object names and property values given by the Data Input process.

4.2.1 Structure of the Automatic Ontology Process

The structure of the Automatic Ontology process is given in Figure 4.1. It shows the five modules and presents information exchange between the modules. The index numbers before information and functions show the information flow with in Automatic Ontology.



Data Input
1.1 object name
1.4 concept name
1.2 object name
1. Concept Creation
1.6 create concept name
1.5 concept name
1.3 concept name
1.8 (2.1) concept name
1.9(2.4) relevant concept names
2.2 concept name
Robot Ontology
2. Relation Creation
2.8 concept names
2.3 relevant concept names and "is-a" relations
2.5 create concept hierarchy
2.10 create non-hierarchy relations
2.9 existing concept names
2.6 concept name
Data and Information Retrieval
2.7 relevant concept names for non-hierarchy relations
2.11(3.1) concept names and relations
3. OWL Creation
3.2 create OWL
3.3 OWL tags
1.7 (4.1) concept name to create instance of concept
4. 4. OWL tags of instance
4. Instance Creation
4.2 create instance
4.3(5.3) instance name
5.1 property value (CDs, JJs)
5.6 OWL tags
5. Property Creation
5.2 classify property name
5.4 create property name and value
5.5 create OWL
Semantic Knowledge Acquisition
Automatic Ontology

Figure 4.1 The structure of the Automatic Ontology process

4.2.2 Concept Creation Module

The Concept Creation module creates the concepts. The main idea is to assume that the object name which is provided by the Data Input process is a concept in WordNet, and to search for the parent of the given concept name in WordNet. Because concepts are arranged in a hierarchy in WordNet, every concept in WordNet must have one or more parents. Therefore, if a parent cannot be found, then the assumption is false; otherwise, the assumption is true. In the case where the given object name's parent is in WordNet and the object name's antecedent is a "physical entity" concept, it is accepted as a concept.

In the case where the parent cannot be found in WordNet, the Concept Creation module will carry on the search into Web Documents. The keyphrase search is employed in the Web Documents search. The keyphrase is formed in the format such as "object name is". The search will return sentences in which object name is a noun. The noun is accepted as the candidate category name to which the object belongs. Then, a calculation on the frequency of the occurrence of the category names is undertaken. The one with the highest frequency is selected as the category name. Concept Creation then sends the category name as the parent of the given object name to WordNet again to see if the parent of the category name is in WordNet and the category name's antecedent is a "physical entity" concept. If the parent of the category name does, then it is accepted as the concept name. If not, Concept Creation will have to choose another noun that has the second highest frequency as a new category name.

Concept Creation will also need to check with Robot Ontology to confirm the concept. In the case where the concept exists in Robot Ontology, Concept Creation sends the concept name and the number of instances to Instance Creation to create instance(s) of the concept. In the case where the concept does not exist in Robot Ontology, the new concept name is retained and then sent to Relation Creation to finalise concept creation. The ConceptCreation algorithm is given in Figure 4.2.

```
Algorithm: ConceptCreation
Input:      O
C ← O;

Begin
    Do {
        A ← SearchParent (C);
        If (A) Then
            B← SearchPhysicalEntity (C);
        If (! A ||! B) Then
            C ←WebSearch (O);
    } While (A==true && B==true)
    SearchRobotOntology (C);
```

Figure 4.2 ConceptCreation algorithm

In this algorithm, O represents object name, C stands for concept, A is the parent of the concept in WordNet and B is the concept that is "physical entity" concept. The algorithm contains two steps. The first is to search for the parent of the concept in WordNet. In the case where the parent of the concept does not exist in WordNet, the algorithm finds the concept name from Web Documents. It is mainly a loop with the condition of while (A==true&&B==true), meaning the concept hierarchy is created from the parent of the concept in WordNet and the concept is a physical entity. This step performs data and information retrieval from WordNet and Web Documents using the following three functions:

- SearchParent () –searching for the parent of the given object name in WordNet
- SearchPhysicalEntity () – searching for the "physical entity" concept, and
- WebSearch () – searching for the category name.

The second step is to determine whether the newly created concept is already in Robot Ontology, implemented by running SearchRobotOntology ().

The SearchParent () function assigns each synset as a level of the hierarchy. This is because WordNet arranges concepts in a hierarchy and English words into synsets and sets of synonyms. In this function, Level 1 is the given object name, Level 2 stores synonyms of the concept name, Level 3 stores the parent(s) of the concept name, and Level 4 stores the parent(s) of the parent(s). This is continued until Level n that stores the top concept of the hierarchy. The search jumps to Level 3 directly from Level 1. In Level 2 synonyms of the concept name are presented. Synonyms do not present the hypernym (a broad meaning), hyponym (a specific meaning), or “is-a” relation. The parents of the concept name are presented in Level 3. If a parent can be found at Level 3, the function returns true, otherwise, it returns false.

SearchPhysicalEntity () carries on the search in the same manner as in the previous function, but at the end of each step of the search, it compares each parent concept with the “physical entity” concept. The function terminates and returns true when they match. Otherwise, the process carries on to Level n and returns false if still not a “physical entity”. Because the concept name must be a tangible and visible object.

WebSearch () works in the following manner: First, “object name” keyword is sent to the search engine in order to search for websites that contain “object name” keyword. Second, Web Documents, the contents in websites, are searched for the key noun by using the keyphrase. The keyphrase is formed in the format of “key noun+ object name+ key noun+ “is” + key noun”. The key nouns are nouns that are obtained from sentences that match the format of the keyphrase and are retrieved from Web Documents. The key noun can be a noun or an empty or zero value. Some sentences cannot be found with a noun for each key noun. Therefore, an empty or zero value can fill a key noun. The key noun will be filled with nouns or an empty or zero value that appear in sentences with the same format retrieved from Web Documents. This function collects URLs of Web Documents that contain the keyphrase, collects the sentences in a URL that contains the keyphrase and counts the frequency of occurrence of the key noun. The function returns the category name which has the highest frequency of the key noun to be the category name of the object name.

SearchRobotOntology () loads all concepts and instances in Robot Ontology by checking the IRI attribute of the concept. The string value of the IRI indicates the same structure if two string values of IRIs are the same. The function returns the concept and the number of instances. In the case where the concept is not in Robot Ontology, the number of instances is zero.

Example 4.1:

Given "cocktail" as the object name, the algorithm assigns "cocktail" to concept_name. It calls SearchParent () to search for the parents of cocktail in Level 3 in the hierarchy (shown in Figure 4.3.) and finds the "course" and "alcohol" concepts. The algorithm continues the search for the "physical entity" concept in higher levels by calling the SearchPhysicalEntity () function. The "physical entity" concept is found at Level 8. The algorithm searches for the "cocktail" concept in Robot Ontology. The SearchRobotOntology () function returns the number of instances. In the case that the "cocktail" concept does not exist in Robot Ontology, the number of instances is zero.

```
Level 1: cocktail
Level 2: mixed_drink, appetizer
Level 3: course, alcohol
Level 4: drug_of_use, nutriment, beverage
Level 5: food, drug, liquid
Level 6: agent, fluid substance,
Level 7: substance, causal_agent, matter
Level 8: physical entity
```

Figure 4.3 Hierarchy of concepts from the "cocktail" to "physical entity" concepts

Example 4.2:

This example shows the case where concept name cannot be found in WordNet. Giving O "plara", the algorithm assigns it to concept_name. The algorithm searches for the parent of "plara" at Level 3. The parent of plara cannot be found in WordNet. Therefore, "plara" is sent to the search engine and the algorithm starts to search for "plara" in Web Documents by calling the WebSearch ()

function. The keyphrase has the format of “key_noun + plara + key_noun + “is” + key_noun”. The algorithm retrieves sentences such as, “Plara is a must for northeastern-tradition dishes, especially som-tam or papaya salad.” and “Som-tam with plara is called som-tam lao (Laotian -style papaya salad) or simply som-tam plara, which is in contrast to the traditional Thai style som-tam, which has dried shrimp and is som-tam thai” etc. Thus, key nouns from Web Documents consist of “dish”, “som-tam”, “papaya”, “salad”, “lao”, “laotian” “style”, “papaya”, “salad”, “thai”, “shrimp” etc. The category name and frequency of occurrence are sorted in descending order as “som-tam” (6), “fish” (5), “papaya” (3), “shrimp” (2) etc. The frequency of occurrence is presented in the parentheses. The highest frequency category name is “som-tam”. The algorithm searches for the parent of “som-tam” at Level 3. The parent of “som-tam” cannot be found in WordNet. Then the algorithm retrieves the next category name as “fish”. The algorithm searches for the parent of “fish” in Level 3. The parent of “fish” can be found in WordNet (shown in Figure 4.4.); then the search continues for the “physical entity” concept at any level. The physical entity concept can be found at Level 4. Therefore, “fish” is accepted as the concept. The algorithm represents the “plara” object name as the “fish” concept. The algorithm searches for the concept in Robot Ontology. The SearchRobotOntology () function returns the number of instances as zero because the “fish” concept cannot be found in Robot Ontology.

Level 1: fish
Level 2: person, aquatic_vertebrate, sign_of_the_zodiac, food
Level 3: vertebrate, region, solid, causal_agent, organism
Level 4: matter, location, living_thing, chordate, physical entity

Figure 4.4 Hierarchy of the “fish” to “physical entity” concepts

4.2.3 Relation Creation Module

The Relation Creation module constructs relations. In the case where the parent concept of the newly created concept exists in Robot Ontology, this module creates the relation between the newly created concept and the existing one in Robot Ontology. In the case where the parent is not an existing concept in Robot

Ontology, this module identifies the relevant concepts and builds up the relations between the newly created concept and the relevant concept until reaching a concept that exists in Robot Ontology. The relevant concepts are selected concepts that appear in the candidate concepts. This module establishes the “is-a” relation using WordNet and “HasA” and “MadeOf” relations using ConceptNet, because WordNet provides hierarchy relations and ConceptNet provides non-hierarchy relations. The ConceptRelation algorithm is given in Figure 4.5.

In this algorithm, C represents concept, CCT stands for concepts from the subset of WordNet from Level 4 (The level contains the parent of the parent of the newly created concept from Concept Creation, to the level which is just below the top level where the “physical entity” concept resides), C stands for concept, P represents parents of the concept, CRT stands for concept relation text, PE stands for physical entity and RC stands for relevant concepts. The relevant concepts will then be identified from the candidate concepts.

The algorithm contains two steps. The first is to create an “is-a” relation between concepts and the parents of concepts or physical entity. It is a loop with the condition of “while (ParentConcept (P) in CCT)”, meaning the parents of the concept are selected from the subset of WordNet from Level 4. The second step is to create “HasA” and/or “MadeOf” relations between the newly created concept that is already in Robot Ontology and the relevant concept, implemented by running CheckRobotOntology ().

```
Algorithm: RelationCreation
Input:      C
Begin
    CCT ← CandidateConcept (C); P ← ParentConcept (C);
    If (CheckRobotOntology (P)) then
    {
        Add "is-a", C, P to CRT;
    }
    Else if P == "PE"
    {
        Add "is-a", C, PE to CRT;
    }
    Else
    {
        Do {
            RC ← P;
            If RC == "PE" or SynonymConcept (RC) == "PE" then
            {
                Add "is-a", C, RC and "is-a", RC, PE to CRT;
            }
            Else
            {
                Add "is-a", C, RC to CRT; C ← RC;
                If (CheckRobotOntology (ParentConcept (P))) then
                {
                    Add "is-a", C, P to CRT;
                }
            }
        } While (ParentConcept (P) in CCT)
    }
    If (CheckRobotOntology (SearchHasA (C))) then
    {
        Add "HasA", C, RC to CRT;
    }
    If (CheckRobotOntology (SearchMadeOf (C))) then
    {
        Add "MadeOf", C, RC to CRT;
    }
Return CRT;
```

Figure 4.5 RelationCreation Algorithm

To implement the first step, assigning the “is-a” relation, the following four functions are used:

- CandidateConcept () – creating a group of concepts
- ParentConcept () – searching for parents of the concept
- SynonymConcept () – searching for the synonyms of the concept
- CheckRobotOntology () – checking if the parent concept or relevant concept is in Robot Ontology

To implement the second step, the following functions are used:

- CheckRobotOntology () – checking if the parent concept or relevant concept is in Robot Ontology
- SearchHasA () – searching for the relevant concept with a “HasA” relation
- SearchMadeOf () – searching for the relevant concept with a “MadeOf” relation.

CandidateConcept () collects candidate concepts from the subset of WordNet from Level 4; the level contains the parent of the parent of the newly created concept from Concept Creation, to the level which is just below the top level where the “physical entity” concept resides. The relevant concept will then be identified from the candidate concepts.

ParentConcept () searches for parents of the newly created concept at Level 3 and returns the parents.

CheckRobotOntology () loads all concept nodes in Robot Ontology and checks the IRI attribute of the concept in question against that of the existing concepts. The function returns true if the IRI attribute of the concept matches that of an existing concept in Robot Ontology. Otherwise, it returns false.

SearchHasA () searches for the relevant concept that has a “HasA” relation with the newly created concept in ConceptNet. The ConceptNet URL is “http://conceptnet5.media.mit.edu/data/5.4/search”. The “HasA&start+ concept” argument is set to specify the “HasA” relation with a start parameter. The start parameter means that the function retrieves the relevant concepts from “concept has a relevant concept” in ConceptNet. In the case where the relevant concept can be found in Robot Ontology, the function adds the “HasA” relation between the newly created concept and the relevant concept.

SearchMadeOf () has the same process as the previous function, but it searches for the relevant concept that has a “MadeOf” relation with the newly created concept in ConceptNet. The start parameter means that the function retrieves the relevant concepts from “concept made of relevant concept” in ConceptNet. In the case where the relevant concept can be found in Robot Ontology, the function adds the “MadeOf” relation between the newly created concept and the relevant concept.

This framework selects the “HasA” and “MadeOf” relations. The “HasA” relation presents the “noun phrase has noun phrase” sentence pattern and the “MadeOf” relation presents the “noun phrase is made of noun phrase” sentence pattern. “HasA” and “MadeOf” relations indicate the meaning between two concepts. For example, the “beer” concept has “HasA” relation with the “alcohol” concept. It means that, beer contains an alcohol. In the case there is no beer concept in the environment, these two relations can be used to implement finding the object that contains alcohol in order to provide a similar object to beer in the environment.

Example 4.3:

Given the “cocktail” concept as a newly created concept, the algorithm calls the CandidateConcept () function to collect the candidate concepts of “cocktail” as shown in Figure 4.6. First, the algorithm searches for the parent of cocktail in Level 3; it finds “course” and “alcohol” by using the ParentConcept () function. Second, the parent of the “cocktail” concept must be checked for existence in Robot Ontology by using the CheckRobotOntology () function. The “course” concept cannot be found in Robot Ontology. The algorithm checks the parent of the “course” concept is not the “physical entity” concept. The parents of “course” are “location”, “social_group”, “food”, “activity”, “ordering”, “path”, “act” and "artefact" by using the ParentConcept () function. The "course" concept is not the "physical entity" concept and synonyms of "course" are found by calling the SynonymConcept () function. The "education", "facility", "gathering", "series", "direction", "layer" and "action" concepts are not the "physical entity" concepts. In the case where the "food" concept can be found in the candidate concepts, the algorithm adds the "is-a" relation between "cocktail" and "course". It assigns "course" as the concept and assigns "food" as the parent concept. The CheckRobotOntology () function returns false for the "food" concept. The algorithm continues to search for the parent of the "food" concept.

The parents of "food" are "cognition" and "matter". In the case where "matter" can be found in the candidate concept, the algorithm assigns "food" as the relevant concept. The "food" concept is not the "physical entity" concept and the synonyms of the "food" concept are not the "physical entity" concept. The algorithm adds the "is-a" relation between the "course" and "food" concepts. The algorithm sets "food" as the concept and sets "matter" as the parent concept. The CheckRobotOntology () function returns false for the "matter" concept. The algorithm continues to search for the parent of the "matter" concept. There is no parent of "matter" in the candidate concept but the synonym of matter is physical entity. The algorithm adds the "is-a" relation between the "food" and "matter" concepts and adds the "is-a" relation between the "matter" and "physical entity" concepts.

Level 4: drug_of_use, nutriment, beverage
Level 5: food, drug, liquid
Level 6: agent, fluid substance
Level 7: substance, causal_agent, matter

Figure 4.6 The candidate concepts for the "cocktail" concept

The algorithm also applies the same process to the parents of "alcohol" which are "drug", "fluid" and "food" in order to create the hierarchy of "cocktail". The SearchHasA and SearchMadeOf functions will be activated after the algorithm creates the hierarchy of "cocktail". The SearchHasA () and SearchMadeOf ()

functions search for the relevant concept name from ConceptNet. They search HasA&start=/c/en/+ cocktail and MadeOf&start=/c/en/+ cocktail, respectively in ConceptNet. In the case where the relevant concept name cannot be found in ConceptNet, the algorithm does not create a relation between cocktail and the relevant concept name, as shown in Figure 4.7. The concept relation text is sent to OWL Creation.

CRT = “is-a”, cocktail, course, “is-a”, course, food, “is-a”, food, matter, “is-a”, matter, “physical entity”, “is-a”, cocktail, alcohol, “is-a”, alcohol, food, “is-a”, alcohol, drug, “is-a”, alcohol, fluid, “is-a”, drug, “physical entity”, “is-a”, fluid, “physical entity”

Figure 4.7 The concept relation text of the “cocktail” concept

Example 4.4:

Given “beer” as a new concept, the parent of beer can be found in WordNet and beer’s antecedent is the “physical entity” concept. The candidate concepts of “beer” are “agent”, “matter”, “food”, “drug_of_use”, “fluid”, “beverage”, “substance”, “causal_agent”, “liquid” and “drug”, by using the CandidateConcept () function. The parent of “beer” is “alcohol” by using the ParentConcept () function. In the case that “alcohol” can be found in Robot Ontology, the algorithm assigns the “is-a” relation between “beer” and “alcohol” into CRT. The SearchHasA () and SearchMadeOf () functions will be activated. The “beer” concept has a “HasA” relation with “water” and “alcohol”. If the “alcohol” concept can be found in Robot Ontology, then the algorithm assigns the “HasA” relation between the “beer” and “alcohol” concepts to concept relation text. The “beer” concept has the “MadeOf” relation with “malt”, which cannot be found in Robot Ontology, so the algorithm does not create the relation between the concepts. The concept relation text of “cocktail” and “beer” is shown in Figure 4.8.

CRT = “is-a”, cocktail, course, “is-a”, course, food, “is-a”, food, matter, “is-a”, matter, “physical entity”, “is-a”, cocktail, alcohol, “is-a”, alcohol, food, “is-a”, alcohol, drug, “is-a”, alcohol, fluid, “is-a”, drug, “physical entity”, “is-a”, fluid, “physical entity”, “is-a”, beer, alcohol, “HasA”, beer, alcohol

Figure 4.8 The concept relation text of the “cocktail” and “beer” concepts

4.2.4 OWL Creation Module

OWL Creation generates OWL tags from the concept relation text that is obtained from Relation Creation. OWL is used to implement an ontology for sharing understanding between human and robots. The OWL tags are stored as part of the Ontology Level in Robot Ontology. Human and robots use Robot Ontology in order to understand objects in a dynamic environment. The OWL Creation process involves converting the concept relation text to OWL tags and updating Robot Ontology. The OWLCreation algorithm is given in Figure 4.9.

Algorithm: OWLCreation

```
Input:      CRT
Begin
    Do
    {
        CreateOWL (Split CRT with comma);
    }
    While (CRT! = End Of File)
```

Figure 4.9 OWLCreation Algorithm

In this algorithm, CRT represents concept relation text. The algorithm splits concept relation text with commas (“,”) in order to create an OWL tag from the relation between the two concepts. The first phrase in the CRT is the relation name and the next two phrases are the child concept and the parent concept, respectively. The process carries on through all phrases until End Of File. End Of File means that algorithm cannot read text from the concept relation text. The main body of this algorithm CreateOWL () create OWL and updates Robot Ontology.

After splitting a phrase, the relation, the child concept and the parent concept are created as OWL by using CreateOWL (). In the case where the relation name is “is-a”, CreateOWL () creates the child concept tag and the parent concept tag and creates the relation tag between the child concept tag and the parent concept tag. The syntax is given below:

```
“<Declaration>
        <Class IRI="concept"/>
</Declaration>
<SubClassOf>
        <Class IRI="child_concept"/>
        <Class IRI="parent_concept"/>
</SubClassOf>”.
```

The function first creates the “Declaration” and “Class” elements. It creates an “IRI” attribute of the “Class” element with the concept value. It then creates the “SubClassOf” and “Class” elements and assigns child_concept and parent_concept as values to “IRI” attributes of the “Class” element.

If the relation name is not “is-a”, CreateOWL () creates non-hierarchy relations between the concepts such as

```
“<Declaration>
      <ObjectProperty IRI="# relation_name "/>
 </Declaration>
<EquivalentClasses>
      <Class IRI="child_concept"/>
        <ObjectSomeValuesFrom>
                <ObjectProperty IRI="relation_name"/>
                <Class IRI="parent_concept"/>
        </ObjectSomeValuesFrom>
</EquivalentClasses>”.
```

The function creates the “Declaration” and “ObjectProperty” elements in order to create the non-hierarchy relation name. It assigns an “IRI” attribute of the “ObjectProperty” element with the relation_name value. The function creates the “EquivalentClasses” and “Class” elements. It assigns an “IRI” attribute of the “Class” element with the child_concept value. It appends the “Class” element as a child element of the “EquivalentClasses” element. The function creates the “ObjectSomeValuesFrom”, “ObjectProperty” and “Class” elements. It assigns an “IRI” attribute of the “ObjectProperty” element with the relation_name value. It assigns an “IRI” attribute of the “Class” element with the parent_concept value. The function appends the “ObjectProperty” and “Class” elements as a child element of the “ObjectSomeValuesFrom” element.

Example 4.5:

Given the concept relation text is “is-a, cocktail, alcohol, is-a, alcohol, fluid, is-a, fluid, physical entity, is-a, beer, alcohol, HasA, beer, alcohol”. The algorithm converts concept relation text to OWL as given in Figure 4.10. First, the algorithm splits concept relation text with commas and CreateOWL () creates OWL until it has read through to the end of the concept relation text.

In the case where the relation name is “is-a”, CreateOWL () represents the “is-a” relation between the “cocktail” and “alcohol” concepts. The “cocktail” concept and the “alcohol” concept are declared. The “Declaration” element is the declared element for creating components. The “Class” element represents the concept by creating an “IRI” attribute as the concept name. The “Class” element is a child element of the “Declaration” element. The “is-a” relation between the “cocktail” concept and “alcohol” concept is created. The “cocktail” concept is a subclass of the “alcohol” concept.

Otherwise, CreateOWL () declares relation name and creates the “HasA” or “MadeOf” relations between the two concepts. CreateOWL () represents the “HasA” relation between the “beer” and “alcohol” concepts. CreateOWL () creates the “Declaration” and the “ObjectProperty” elements. The

“ObjectProperty” element represents the relation by creating an “IRI” attribute as the relation name. The “ObjectProperty” element is a child element of the “Declaration” element.

The “EquivalentClasses” and “Class” elements are created. The “EquivalentClasses” element indicates two classes which are equivalent. Two classes are considered equivalent if they contain exactly the same individuals. The “Class” element represents the concept by creating an “IRI” attribute as the “beer” concept. The “Class” element is a child element of the “EquivalentClasses” element. The “ObjectSomeValuesFrom”, “Class” and “ObjectProperty” elements are created. The “ObjectSomeValuesFrom” element has some values in the “Class” element in the “ObjectProperty” element. The “ObjectProperty” element is the relation between the concepts. The “ObjectProperty” element represents the relation by creating an “IRI” attribute as a “HasA” relation. The “Class” element represents the concept by creating an “IRI” attribute as the “alcohol” concept. The “ObjectProperty” and “Class” elements are child elements of the “ObjectSomeValuesFrom” element. The ObjectSomeValuesFrom” element is a child element of the “Class” element. The “Class” element is a child element of the “EquivalentClasses” element.

```
<Declaration>
      <Class IRI="#cocktail"/>
</Declaration>

<Declaration>
      <Class IRI="#alcohol"/>
</Declaration>

<Declaration>
      <Class IRI="#fluid"/>
</Declaration>

<Declaration>
      <Class IRI="#physical entity"/>
</Declaration>

<Declaration>
      <Class IRI="#beer"/>
</Declaration>
```

```
<SubClassOf>
      <Class IRI="#cocktail"/>
      <Class IRI="#alcohol"/>
</SubClassOf>

<SubClassOf>
      <Class IRI="#alcohol"/>
      <Class IRI="#fluid"/>
</SubClassOf>

<SubClassOf>
      <Class IRI="#fluid"/>
      <Class IRI="#physical entity"/>
</SubClassOf>

<SubClassOf>
      <Class IRI="#beer"/>
      <Class IRI="#alcohol"/>
</SubClassOf>
```

```
<Declaration>
      <ObjectProperty IRI="#HasA"/>
</Declaration>
<EquivalentClasses>
      <Class IRI="#beer"/>
         <ObjectSomeValuesFrom>
                  <ObjectProperty IRI="#HasA"/>
                  <Class IRI="#alcohol"/>
         </ObjectSomeValuesFrom>
</EquivalentClasses>
```

Figure 4.10 Example of the OWL syntax of concept relation text

4.2.5 Instance Creation Module

Instance Creation builds instances of a concept. It creates an instance of a concept in OWL format and sends it to Robot Ontology. The InstanceCreation algorithm, as given in Figure 4.11, receives the concept name. The main idea is to create the instance name by using the instance name pattern in OWL format. The instance name pattern is a string format that attaches a “#” symbol, a concept name, a “_” symbol and an instance number.

```
Algorithm: InstanceCreation
Input:      C
Begin
        I←Increase number of instance by 1 and convert to string;
        DeclarationInstance (I);
        CreateInstance (C, I);
```

Figure 4.11 InstanceCreation Algorithm

In this algorithm, C represents concept and I stands for instance. The algorithm has the following functions:

- DeclarationInstance () – creating instance in OWL syntax as

```
<Declaration>
     <NamedIndividual IRI="instance_name"/>
</Declaration>.
```

- CreateInstance () – creating OWL syntax as

```
<ClassAssertion>
     <Class IRI="concept_name"/>
     <NamedIndividual IRI="instance_name"/>
</ClassAssertion>.
```

DeclarationInstance () creates the “Declaration” and “NamedIndividual” elements. It creates an “IRI” attribute of the “NamedIndividual” element with the instance_name value. It appends “NamedIndividual” element to be a child element of the “Declaration” element and appends the “Declaration” element to the instance_tag.

CreateInstance () creates the “ClassAssertion” element and “Class” elements. It creates an “IRI” attribute of the “Class” element with the concept_name value. It creates the “NamedIndividual” element and assigns an “IRI” attribute with the instance_name value. It appends the “Class” and “NamedIndividual” elements to be child elements of the “ClassAssertion” element and appends the “ClassAssertion” element to the instance_tag.

Example 4.6:

Concept Creation sends cocktail as the concept name and zero as the number of instances. First, the algorithm increases the number of instances by 1 and converts the number of instances to a string. It forms the concept name by adding a “#” symbol in front of the concept name and forms the instance name by adding a “#” symbol in front of the instance name and adds a “_” symbol and the number of instances as string. The instance name is “#cocktail_1”. The algorithm calls the DeclarationInstance () function in order to declare the instance name. The result of the DeclarationInstance () function is:

```
“<Declaration>
    <NamedIndividual IRI="#cocktail_1"/>
</Declaration>”.
```

The DeclarationInstance () function creates the “Declaration” and “NamedIndividual” elements. It creates an “IRI” attribute of the “NamedIndividual” element as “#cocktail_1”. The “NamedIndividual” element is a child element of the “Declaration” element.

The algorithm calls the CreateInstance () function in order to create the instance name. It creates the “ClassAssertion”, “Class” and “NamedIndividual” elements. The “Class” and “NamedIndividual” elements are child elements of the “ClassAssertion” element. It creates an “IRI” attribute of the “Class” element as the “cocktail” concept name. It creates an “IRI” attribute of the “NamedIndividual” element as the “#cocktail_1” instance name. The result of the CreateInstance () function is:

```
“<ClassAssertion>
        <Class IRI="#cocktail"/>
        <NamedIndividual IRI="#cocktail_1"/>
</ClassAssertion>”.
```

The result of the instance_tag is given in Figure 4.12.

```
<Declaration>
        <NamedIndividual IRI="#cocktail_1"/>
</Declaration>
<ClassAssertion>
        <Class IRI="#cocktail"/>
        <NamedIndividual IRI="#cocktail_1"/>
</ClassAssertion>
```

Figure 4.12 The OWL tag instance of cocktail

4.2.6 Property Creation Module

Property Creation constructs the property names and their values for the instance from Instance Creation. Property Creation receives the instance name and instance tag from Instance Creation and receives property values from Data Input. The main idea is to classify property names from property values, to assign their property values to each property name, to convert property names and their values to OWL format and to append these to the instance tag.

The PropertyCreation algorithm as given in Figure 4.13 calls the ClassifyProperty () function with property value and calls the AssignProperty () function with instance name, property name and property value. It adds each property value to property tag until the last property value is reached. The algorithm appends instance tag and property tag to owl tag in order to send it to Semantic Knowledge Acquisition.

```
Algorithm: PropertyCreation
Input:        IN, IT, PV
Begin
    IN←Increase number of instance by 1 and convert to string;
    Do
    {
        P [] ← ClassifyProperty (PV [i]);
        PV[i] ←AssignProperty (IN, P[i], PV [i]);
    }
    While (End of PV)
    Create OWL from IT and PT;
    OWL is sent to Semantic Knowledge Acquisition;
```

Figure 4.13 PropertyCreation Algorithm

In this algorithm, IN stands for instance, P stands for property of instance, IT stands for instance text and PV stands for property value. The main loop with the condition of while (End of PV) means creating a property value of instance until reaching the last property value of the instance. The algorithm has the following functions:

- ClassifyProperty () – classifying property value to property name
- AssignProperty () – creating OWL syntax as

```
<DataPropertyAssertion>
    <DataProperty IRI="#property_name"/>
    <NamedIndividual IRI="#instance_name"/>
    <Literal datatypeIRI="&rdf;PlainLiteral">property_value</Literal>
</DataPropertyAssertion>.
```

The ClassifyProperty () function classifies property value to property name by using rules. Adjectives give property values of physical objects from Data Input. There are five predefined properties: “hasTime”, “hasLocation”, “hasColour”, “hasShape” and “hasProperty”. The function matches property value to the predefined properties by using the rules for classifying the property name. The three rules, respectively, are:

Rule1 is defined by a number that has 14 digits. The algorithm assigns “hasTime” to property_name.

Rule2 is defined by text that is separated by “,” symbols. The algorithm assigns “hasLocation” to property_name. Otherwise, the algorithm assigns “hasProperty” to property_name.

AssignProperty () creates the “DataPropertyAssertion” and “DataProperty” elements. It creates an “IRI” attribute of the “DataProperty” element with property_name as its value. The function creates the “NamedIndividual” element and assigns an “IRI” attribute with instance_name as its value. It assigns the “Literal” element, assigns a “datatypeIRI” attribute with “&rdf;PlainLiteral” value and assigns property value to the “Literal” element. Finally, the function appends “DataProperty”, “NamedIndividual” and “Literal” elements to be child elements of the “DataPropertyAssertion” element.

Example 4.7:

Given the property values of the “cocktail_1” instance as “20150819113959” and “1, 0, 1”. The algorithm calls the ClassifyProperty () function with “20150819113959” property value. The ClassifyProperty () function assigns the property name as “hasTime”. The algorithm calls the AssignProperty () function with the “cocktail_1” instance, “hasTime” property name and “20150819113959” property value. The AssignProperty () function creates OWL as property value as:

```
<DataPropertyAssertion>
    <DataProperty IRI="#hasTime"/>
    <NamedIndividual IRI="#cocktail_1"/>
    <Literal datatypeIRI="&rdf;PlainLiteral">20150819113959</Literal>
</DataPropertyAssertion>.
```

The algorithm appends the property value to property tag. The process continues by calling the ClassifyProperty () function with “1, 0, 1” value and it returns property name as “hasLocation”. It calls the ClassifyProperty () function with the "cocktail_1" instance, "hasLocation" property name and "1, 0, 1" property value. The AssignProperty () function creates OWL as property value as:

```
<DataPropertyAssertion>
    <DataProperty IRI="#hasLocation"/>
       <NamedIndividual IRI="#cocktail_1"/>
              <Literal datatypeIRI="&rdf;PlainLiteral">1, 0, 1</Literal>
</DataPropertyAssertion>.
```

The algorithm appends property value to property tag again. The algorithm stops processing when the last property value is reached. The instance tag of "cocktail_1" as shown in Figure 4.12 appends with property tag as OWL. The owl tags as shown in Figure 4.14 is sent to Semantic Knowledge Acquisition.

```
<DataPropertyAssertion>
     <DataProperty IRI="#hasTime"/>
     <NamedIndividual IRI="#cocktail_1"/>
     <Literal datatypeIRI="&rdf; PlainLiteral">20150819113959</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
    <DataProperty IRI="#hasLocation"/>
       <NamedIndividual IRI="# cocktail_1"/>
              <Literal datatypeIRI="&rdf; PlainLiteral">1, 0, 1</Literal>
 </DataPropertyAssertion>
```

Figure 4.14 The instance of cocktail, property names and values

4.3 Summary

The Automatic Ontology process within the DIRAOF framework is presented in order to create concepts, relations, instances of concepts, properties and their values by using WordNet, ConceptNet and Web Documents from Data and Information Retrieval. The Automatic Ontology process consists of five modules: Concept Creation, Relation Creation, OWL Creation, Instance Creation and Property Creation. The studies on the Automatic Ontology process and the Data and Information Retrieval process show that the retrieved concept names and the “is-a”, “HasA” and “MadeOf” relations are created in Robot Ontology. Instances of concepts, properties and their values are created and are sent to the Semantic Knowledge Acquisition. Therefore, Automatic Ontology is able to create the components of the ontology as OWL tags and send them to the Robot Ontology and Semantic Knowledge Acquisition processes.

The Semantic Knowledge Acquisition process is presented in Chapter 5 in order to apply the OWL from Property Creation for representing the relation between instances of a concept and their properties in the environment.

CHAPTER 5 SEMANTIC KNOWLEDGE ACQUISITION

Semantic knowledge refers to knowledge about instances of concepts, properties and property values. From the ontology development point of view, knowledge is represented using an ontology. An ontology provides a formal and explicit specification of a shared conceptualization. It does not contain information about instances in environments. Homecare robots, however, require such information when they are asked to find instances in domestic environments. Therefore, there is a need for associating semantic knowledge with the instances stored in the ontology.

Furthermore, home environments are dynamic and unstructured because human users and robots share the same space. Robots may place objects in one place and human users can move the objects to different places later without informing the robots. Humans can also introduce objects into the space. Semantic knowledge would contribute to robots being able to locate objects in dynamic and unstructured environments.

The Semantic Knowledge Acquisition process within the DIRAOF framework is designed to associate semantic knowledge for instances and to use semantic knowledge for tracing instances. This chapter details the Semantic Knowledge Acquisition process. The structure of Semantic Knowledge Acquisition consists of four modules: Ontology Updating, Semantic Representation, NFile and Object Prediction. This chapter also presents Tracking Instances: that is, a novel method to solve the problem of locating instances in a dynamic environment by predicting the future locations of instances.

5.1 Association of Semantic Knowledge with Instances

Semantic knowledge is available from the Data Input process. The Data Input process retrieves information from Gazebo which is a 3D simulator and simulates populations of robots, objects and sensors. The framework assumes that the labels of objects from Gazebo are retrieved in text-file format.

The label (name) of an object is tackled by Concept Creation and Relation Creation to create concepts, and relations between concepts. An instance is built by Instance Creation after the new concept is created or the new instance of a concept occurs, whereas a property value consists of time value, location value and other property values. They are identified by Property Creation in Automatic Ontology. Property Creation aims to assign a property name for each property value.

It is necessary to associate semantic knowledge with the instance. Thus, Ontology Updating associates semantic knowledge of an instance with the instance in Robot Ontology. Semantic knowledge consists of property names and property values of the instance. There are at least two property names of an instance ("hasLocation" and "hasTime") that always occur for every instance.

5.1.1 Structure of the Semantic Knowledge Acquisition Process

The structure of the Semantic Knowledge Acquisition process is given in Figure 5.1. It consists of four modules: Ontology Updating, Semantic Representation, NFile and Object Prediction. The functionality of each module is described as follows. The Ontology Updating module updates semantic knowledge of instances into Robot Ontology. Semantic Representation uses semantic knowledge of instances to identify FileN, assigns semantic knowledge of instances to FileN and searches for instances of similar concepts to the concept name. The NFile module uses semantic knowledge of instances to calculate the weight function of all instances of a concept and finds instances in the dynamic environment. The Object Prediction module uses semantic knowledge of instances to compute the decision score for predicting the location of the instance of the concept and to send similar instances to the Query process.

Figure 5.1 presents information exchange between Automatic Ontology, Semantic Knowledge Acquisition, Robot Ontology, Result Evaluation and Query. Concepts and relations between concepts from OWL Creation in Automatic Ontology are stored at the Ontology Level in Robot Ontology. Whereas instances, their property names and property values from Property Creation in Automatic Ontology are represented as semantic knowledge by the Semantic Knowledge Acquisition process and they are stored at the Semantic Knowledge Level in Robot Ontology. Query receives the concept name from Robot Ontology and sends it to Semantic Representation. Semantic Knowledge Acquisition manages semantic knowledge of all instances of the concept name and sends the prediction location of the concept name and instances of concept back to Query.

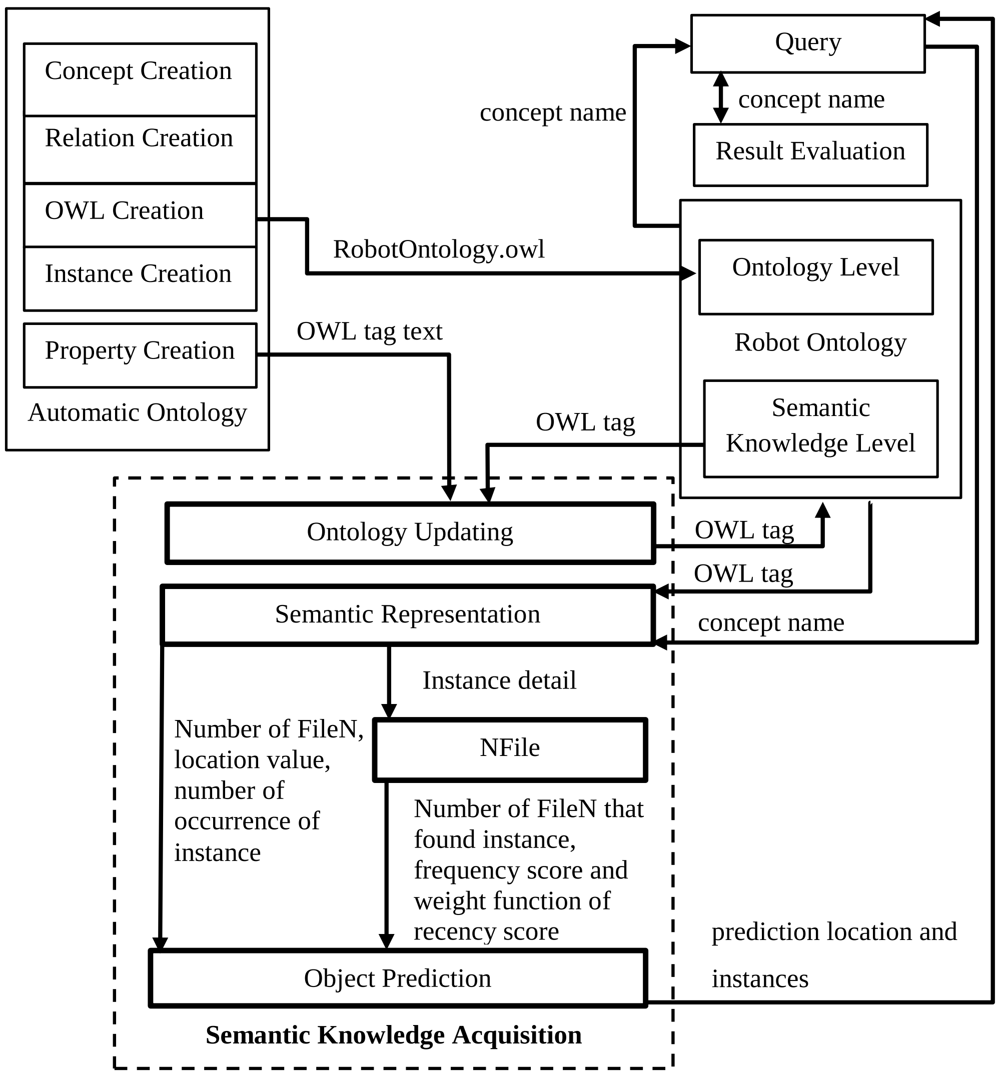


Figure 5.1 The structure of Semantic Knowledge Acquisition

5.1.2 Assigning Semantic Knowledge to the "hasLocation" Property

Semantic knowledge of an instance provides knowledge about the instance in the unstructured environment. The problem of locating objects in dynamic and unstructured environments is tackled by the idea of assigning semantic knowledge of location to the instance of a concept. As mentioned earlier, every instance has a "hasLocation" property name and property value in order to indicate the location of the instance in the environment. The "hasLocation" property value is assigned with semantic knowledge of the location of the instance and is updated into Robot Ontology. The OntologyUpdating algorithm is given in Figure 5.2.

```
Algorithm: OntologyUpdating
Input:          OT
RO ← document is loaded from "RobotOntology.owl";

Begin
        Do {
                UpdateOntology (RO);
        } While (P and PV! = End Of File)
        Return RO;
```

Figure 5.2 OntologyUpdating algorithm

In this algorithm, OT represents an OWL tag as text from Property Creation, RO stands for the Robot Ontology document that stores concepts, relations between concepts, instances, properties and their property value, P stands for property of instance and PV stands for property value. The main loop with the condition of while (P and PV! = End Of File) means updating the property value of an instance until reaching the last property and property value of that instance.

The algorithm has the following function:

- UpdateOntology () – appending OWL tags to RobotOntology.owl

UpdateOntology () updates OWL tag text from Property Creation as semantic knowledge of each instance into Robot Ontology. The OWL tags provide semantic knowledge about instances, property names and their property values. The algorithm retrieves and updates semantic knowledge of instance to Robot Ontology.

Example 5.1:

OWL tag text from Property Creation is

```
“<Declaration>
        <NamedIndividual IRI="#cocktail_2"/>
</Declaration>
<ClassAssertion>
        <Class IRI="#cocktail"/>
        <NamedIndividual IRI="#cocktail_2"/>
</ClassAssertion>
<DataPropertyAssertion>
        <DataProperty IRI="#hasTime"/>
        <NamedIndividual IRI="#cocktail_2"/>
        <Literal datatypeIRI="&rdf; PlainLiteral">20150919113959</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
       <DataProperty IRI="#hasLocation"/>
           <NamedIndividual IRI="#cocktail_2"/>
                    <Literal datatypeIRI="&rdf; PlainLiteral">1, 1, 1</Literal>
</DataPropertyAssertion>”.
```

The OWL tag contains the “cocktail_2” instance, the “hasLocation” and “hasTime” property names and their values. The algorithm appends the OWL tag text that describes the “cocktail_2” instance to Robot Ontology. The OWL document contains concepts, relations, instances, properties and their values. In this case, the “cocktail” concept has one instance, namely the “cocktail_1” instance.

The UpdateOntology () appends the OWL tag of the “cocktail_2” instance to the OWL document as given in Figure 5.3. Figure 5.3 shows concepts, relations, instances, properties and values which are appended by the OWL tag (the italic text). There are two instances of cocktail in the OWL document at this point. Ontology Updating continually receives the OWL tag and updates the OWL document by using the UpdateOntology () until the last property name and property value from Property Creation are reached. The OWL document is updated into RobotOntology.owl.

```
<Declaration>
      <Class IRI="#cocktail"/>
</Declaration>
<Declaration>
      <Class IRI="#alcohol"/>
</Declaration>
<Declaration>
      <Class IRI="#fluid"/>
</Declaration>
<Declaration>
     <Class IRI="#physical entity"/>
</Declaration>
<Declaration>
      <Class IRI="#beer"/>
</Declaration>
```

```
<SubClassOf>
      <Class IRI="#cocktail"/>
      <Class IRI="#alcohol"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#alcohol"/>
      <Class IRI="#fluid"/>
</SubClassOf>
<SubClassOf>
     <Class IRI="# fluid"/>
     <Class IRI="#physical entity"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#beer"/>
      <Class IRI="#alcohol"/>
</SubClassOf>
```

```
<Declaration>
      <ObjectProperty IRI="#HasA"/>
</Declaration>
<EquivalentClasses>
      <Class IRI="#beer"/>
         <ObjectSomeValuesFrom>
                  <ObjectProperty IRI="#HasA"/>
                  <Class IRI="#alcohol"/>
         </ObjectSomeValuesFrom>
</EquivalentClasses>
```

```
<Declaration>
        <NamedIndividual IRI="#cocktail_1"/>
</Declaration>
<ClassAssertion>
        <Class IRI="#cocktail"/>
        <NamedIndividual IRI="#cocktail_1"/>
</ClassAssertion>
<DataPropertyAssertion>
        <DataProperty IRI="#hasTime"/>
        <NamedIndividual IRI="#cocktail_1"/>
        <Literal datatypeIRI="&rdf; PlainLiteral">20150819113959</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
       <DataProperty IRI="#hasLocation"/>
            <NamedIndividual IRI="#cocktail_1"/>
                        <Literal datatypeIRI="&rdf; PlainLiteral">1, 0, 1</Literal>
</DataPropertyAssertion>
<Declaration>
        <NamedIndividual IRI="#cocktail_2"/>
</Declaration>
<ClassAssertion>
        <Class IRI="#cocktail"/>
        <NamedIndividual IRI="#cocktail_2"/>
</ClassAssertion>
<DataPropertyAssertion>
        <DataProperty IRI="#hasTime"/>
        <NamedIndividual IRI="#cocktail_2"/>
        <Literal datatypeIRI="&rdf; PlainLiteral">20150919113959</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
       <DataProperty IRI="#hasLocation"/>
            <NamedIndividual IRI="# cocktail_2"/>
                        <Literal datatypeIRI="&rdf; PlainLiteral">1, 1, 1</Literal>
</DataPropertyAssertion>
```

Figure 5.3 The OWL document is updated by OWL tags

5.2 Tracking Instances

An instance has a semantic knowledge of location at one time given by the “hasTime” property value. Thus, semantic knowledge of location can be traced from time to time. This idea of assigning location of semantic knowledge leads to trace instances for predicting the future location of instance.

Tracking Instances presents the method for locating instances with semantic knowledge. Tracking Instances gathers semantic knowledge from instances and predicts the future location of instance. It uses semantic knowledge to predict the future location of an instance. NFile, Recency and Frequency techniques are applied in Tracking Instances.

5.2.1 NFile Module

The NFile module computes the summation of the weight function of each instance of a concept name. NFile module searches for number of FileN that instance occur at specific location. FileN is a system that contains the number of files. Each file records the location of an object during a period of time. FileN is created in the Semantic Representation module.

Each instance of the concept name that appears in FileN is calculated by the weight function. The pair of weight functions is presented in equation 5.1a and equation 5.1b.

$$F_{\text{frequency}}(x) = 1 \qquad (5.1a)$$

$$F_{\text{recency}}(x) = 0.5^{(\text{NFileN}-x)} \qquad (5.1b)$$

$F(x)$ is a weight function. X is the index of the file where a particular concept is found in NFileN. NFileN is the total number of FileN files. There are two types of weight function: weight function of frequency (equation 5.1a) and weight function of recency (equation 5.1b). The weight function of frequency counts instances of the same concept as 1 for each file. Instances represent the concept and they can change from place to place in a dynamic environment. The concept represents concrete objects that tend to be placed in a specific location. The weight function of recency is (0.5) to the power the number of total FileN files minus the number of FileN of that instance which was found in the location.

The NFile algorithm is given in Figure 5.5.

```
Algorithm: NFile
Input:      RO
FC ← 1;

Begin
        Do {
                FF ← Find_FileN (RO);
                If (FF) Then
                        WF=WeightF (FF);
                        WR=WeightR (FF);
        } While (FC! = Max (FC))
        Return FF, WF, WR
```

Figure 5.4 NFile algorithm

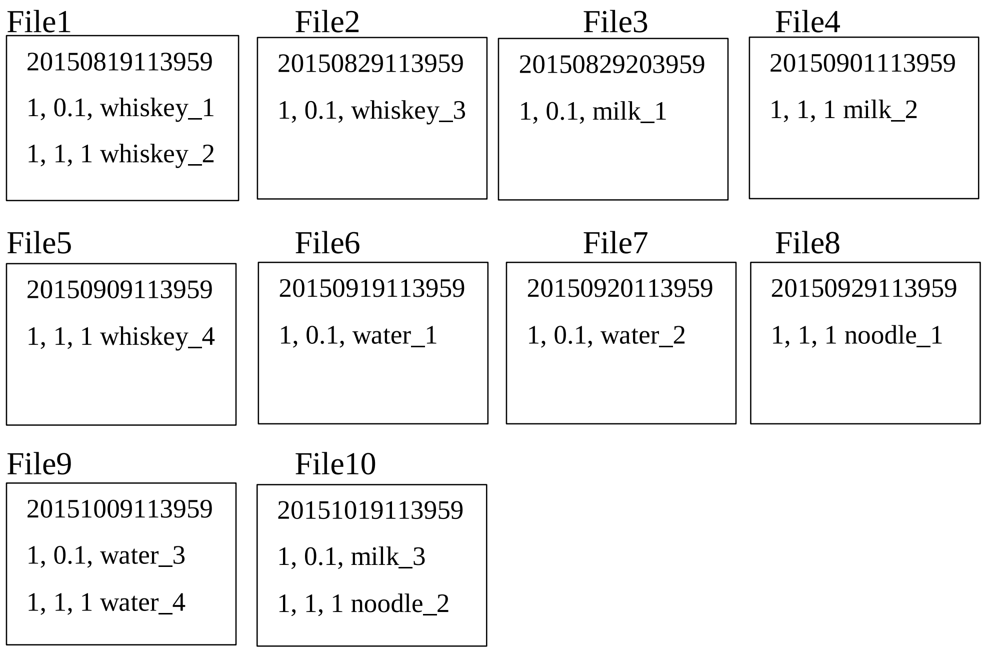
In this algorithm, RO represents Robot Ontology, FC stands for the number of FileN, FF represents the number of the file of that instance found, WF stands for the weight function of frequency and WR stands for the weight function of recency. The main loop with the condition of while (FC! = Max (FC) means processing the WeightF and WeightR functions until the maximum number of FileN. The algorithm uses the following three functions:

- Find_FileN () – searching for FileN of each instance name
- WeightF () – calculating the summation of the weight function for the frequency score
- WeightR () – calculating the summation of the weight function for the recency score.

Find_FileN () searches for FileN in Robot Ontology. The number of FileN is used to compute the summation of the weight function for the frequency score and recency score.

WeightF () calculates the summation of the weight function for the frequency score from equation 5.1.

WeightR () calculates the summation of the weight function for the recency score from equation 5.1.


File1
20150819113959
1, 0.1, whiskey_1
1, 1, 1 whiskey_2

File2
20150829113959
1, 0.1, whiskey_3

File3
20150829203959
1, 0.1, milk_1

File4
20150901113959
1, 1, 1 milk_2

File5
20150909113959
1, 1, 1 whiskey_4

File6
20150919113959
1, 0.1, water_1

File7
20150920113959
1, 0.1, water_2

File8
20150929113959
1, 1, 1 noodle_1

File9
20151009113959
1, 0.1, water_3
1, 1, 1 water_4

File10
20151019113959
1, 0.1, milk_3
1, 1, 1 noodle_2

Figure 5.5 The number of NFileN is 10

Figure 5.5 shows how to calculate weight function of instances that occur in a period of 10 files of FileN. Each file consists of times locations and instances. There are four instances of whiskey as given in Figure 5.4. There are two instances of whiskey that occur in File1 (whiskey_1 and whiskey_2). File2 found whiskey_3 instance, and File5 found whiskey_4 instance. The weight function of frequency of whiskey concept is 3. The weight function of recency of whiskey concept can be calculated from f (5) = (0.5) $^{(10-5)}$.

5.2.2 Frequency and Recency

Dynamic environments are random and they cannot guarantee the future location of instances. Randomness is an unstructured event that can give results in any of several outcomes. It cannot predict the outcome in any particular case. Frequency can be counted in the randomness. Instances occur in lawless locations in FileN. The locations have an equal chance of getting sampled. Selecting one particular location does not affect the chances of any other location being selected. Semantic knowledge of instances can be used by counting the frequency of a particular instance in the same location in FileN. The frequency is counted and calculated for locating the instance. The frequency known as spatial locality that an object will appear again based on how often it has been seen before. Frequency counts do not pertain only to the lifespan of a particular object, but can also be persistent across multiple lifetimes of the object.

Frequency relates to location that human often places the object. However, human can change their behaviour or the furniture can be moved from place to place in dynamic environment. Robot may not be able to find the object. Recency should be included.

Recency means the latest time that robot makes a scan to recognise objects. Recency is applied in order to gather the recent location of instances. Recency is one of theory and observation of human memory. It refers to the decrease of memory performance with the time since an instance was presented. The last instance is remembered much better than the previous instances. The Tracking Instances keeps track of the recent instance in order to gain the number of FileN. Because the recent FileN is remembered, it tends to find the instance.

Tracking Instances is implemented in random experiments. A random experiment is repeated multiple times under the same conditions. It shows possible results but cannot show precise results, because the Tracking Instances will be implemented in real environments where results cannot be guaranteed. The sample space presents the set of possible outcomes of random experiments. Thus, the sample space is all locations in the environment.

The key problem is to find out the location of an object in a dynamic environment. The meaning of finding an object is about the location of the object. The location is related to Frequency and Recency. Frequency indicates how many times that robot found the object in a particular location. Recency indicates the latest location of the object in FileN.

Object Prediction applies Frequency and Recency for computing the decision score. Object Prediction predicts the location of an instance. Object Prediction calculates all decision scores of each location of the instances. Because the same instances occur in same FileN and robots do not classify the same instance of a concept in different scenes. The result of Object Prediction is the highest decision score of location.

The weight function is used to balance the decision score of location by multiplying the weight function of frequency with the frequency score and multiplying the weight function of recency with the recency score. The frequency score is calculated from the number of times that the robot found the instance in a particular location divided by the number of total FileN files. The recency score calculates from the latest file number divided by the number of total FileN files. The decision score of location is presented in equation 5.2.

$$D_{loc} = \frac{F_{loc}}{NFileN} + \left((0.5)^{NFileN-RFileN} \times \frac{RFileN}{NFileN} \right) \qquad (5.2)$$

D_{loc} is the decision score of the specific location (loc). F_{loc} is the number of times that the robot found the instance in a particular location. NFileN is the total number of FileN files. RFileN is the latest file number that the robot found an instance in that location. The RFileN value can range in value from 1 to NFileN.

O notation (Mehlhorn and Sanders, 2008) is used to describe the performance or complexity of an algorithm. It describes the execution time required by an algorithm. Equation 5.2 is used by Tracking Instances to calculate the decision score for each location. Equation 5.2 is justified by O notation in order to describe the performance of this equation for calculating the decision score of each location that the instance was found. O notation gives time complexity (CPU usage). The time required by equation 5.2 gives $\mathrm{O}(n)$ time complexity as follows:

$$\mathrm{O}(\frac{F_{loc}}{NFileN}) + O((0.5)^{NFileN-RFileN}) \times \mathrm{O}(\frac{RFileN}{NFileN})$$

$\mathrm{O}(\frac{F_{loc}}{NFileN})$ gives $\mathrm{O}(n)$ time complexity,

$\mathrm{O}((0.5)^{NFileN-RFileN})$ gives $\mathrm{O}(1)$ time complexity and

$\mathrm{O}\left(\frac{RFileN}{NFileN}\right)$ gives $\mathrm{O}(n)$ time complexity.

Thus, equation 5.2 gives $\mathrm{O}(n)$ time complexity for calculating the prediction location of an instance. $\mathrm{O}(n)$ shows that equation 5.2 will grow linearly and in direct proportion to the size of the input dataset.

The ObjectPrediction algorithm is given in Figure 5.6.

```
Algorithm: ObjectPrediction
Input:        FN, LI, FF, WF, WR

Begin
    Do {
        PS ← PredictionScore (FN, FF, WF, WR);

    } While (LI! = Last instance location)
    If (Max of PS) Then
    {
        PL ← IL of Max of PS;
    }
    Return PL;
```

Figure 5.6 ObjectPrediction algorithm

In this algorithm, FN represents the total number of FileN, LI stands for location of instance, FF represents the number of FileN of that instance found, WF stands for weight function of frequency, WR stands for weight function of recency, PS stands for prediction score, IL stands for instance location and PL represents prediction location.

The main loop with the condition of while (LI! = Last instance location) means calculating the prediction score until reaching the last instance location. The maximum value of the prediction score is selected as the prediction location.

The algorithm has the following function:

- PredictionScore () – calculating prediction score following equation 5.2.

Example 5.2:

The total of FileN is 2 as shown in Figure 5.3. There are two locations of the “cocktail” instance: “1, 0, 1” and “1, 1, 1”. The number of FileN that found the “cocktail_1” instance is 1. The number of FileN that found the “cocktail_2” instance is 2. The decision score of “1, 0, 1” location is calculated as (1/2) + $((0.5)^{2-1} * (1/2)$, which equals 0.75. The decision score of “1, 1, 1” location is calculated as (1/2) + $((0.5)^{2-2} * (2/2)$, which equals 1.5. ObjectPrediction selects the maximum value of the prediction score as 1.5. Therefore, the prediction location is “1, 1, 1”.

5.3 Usage of Semantic Knowledge Based on Query

After semantic knowledge of instances is associated with the instances and assigned to instances in Robot Ontology, the Query process inquires the future locations of instances and the similar objects. First, the future location of instance is queried by calculating the decision location of instance. Second, searching for similar objects in Robot Ontology is needed in order to present the similar objects in dynamic environment where the robot cannot find the particular object.

The ideas for predicting locations are

- grouping the same semantic knowledge of the “hasTime” property name in order to identify FileN

- grouping the same semantic knowledge of the “hasLocation” property name of instances of concept in order to count the frequency and to calculate the weight function of recency
- applying Tracking Instances to calculate the prediction score of each location and
- selecting the highest prediction score of location as the future location of instance.

Semantic Representation manages instances of a concept by using semantic knowledge, due to the framework automatically querying the concept name from the Query process. Semantic Representation receives the concept name from Query and receives the updated RobotOntology.owl from Robot Ontology. Semantic Representation searches for instances of the concept name, sorts them by the “hasTime” value, assigns FileN to each instance of the concept and counts the “hasLocation” value of the instance.

The algorithm searches for instances of the concept name in Robot Ontology. In the case where instances of the concept name can be found, it returns all instance names of the concept name. The algorithm sorts the “hasTime” property values of instances, assigns FileN to each instance and counts the occurrence of “hasLocation” for each instance.

The “hasTime” property value indicates each time that the robot observes approaches the objects. One “hasTime” property value is a FileN which starts from 1. FileN is increased by 1 for each “hasTime” property value. FileN is the time period from the reference in the past to the current time. Because there are instances, property names and values occur in a FileN.

The ideas for searching for similar objects are

- searching for parent concepts of the concept
- searching for similar concepts by finding “HasA” and/or “MadeOf” relations with the parent concept name in Robot Ontology
- searching for instances of the similar concepts in the recent FileN.

```
Algorithm: SemanticRepresentation
Input:     C
RO ← document is loaded from "RobotOntology.owl";

Begin
        SortHasTime (RO);
        Do {
                AssignFileN (SearchInstance (C), count the "hasTime" value of all
instances in RO);
                CountLocation ("hasLocation" value of all instances of C);
                SearchSimilar (C);
        } While (IN! =last IN of C)
```

Figure 5.7 SemanticRepresentation algorithm

The SemanticRepresentation algorithm is given in Figure 5.7. In this algorithm, C represents concept name, RO stands for the Robot Ontology document that stores concepts, relations between concepts, instances, properties and their property values and IN stands for instance name. The main loop with the condition of while (IN! = last IN of C) means processing from the first instance name until reaching the last instance name of each concept.

The algorithm has the following functions:

- SortHasTime () –sorts the "hasTime" property values
- AssignFileN () – assigns the number of FileN to the instance name
- SearchInstance () – searches for the instance name of the concept in the OWL document
- CountLocation () – counts and assigns the value of the "hasLocation" property of all instances of the concept name
- SearchSimilar () – searches for similar concepts by finding the similar concepts that have "HasA" and/or "MadeOf" relations with the concept name in Robot Ontology.

SortHasTime () sorts the "hasTime" property values. There are "hasTime" property values in Robot Ontology. The "hasTime" property values are sorted by ascending order and duplicate values are eliminated. The function searches for the IRI attribute of the “DataProperty” element. The “DataProperty” element is a child element of the “DataPropertyAssertion” element as:

```
“<DataPropertyAssertion>
        <DataProperty IRI="property name"/>
        <NamedIndividual IRI="instance name"/>
        <Literal datatypeIRI="&rdf;PlainLiteral">property value</Literal>
</DataPropertyAssertion>” OWL tag.
```

In the case where the IRI attribute of the “DataProperty” element is “hasTime”, the function gets the “Literal” element value that is the same child element of the “DataPropertyAssertion” element. First, it starts to assign the FileN value as 1. Second, FileN is increased by 1 in order to sort the “Literal” element values in ascending order and eliminate duplicate values. It continues the process for all “Literal” element values until End Of File.

AssignFileN () assigns the number of FileN to the instance name. An instance appears in a FileN. A FileN contains many instances. The value of “hasTime” of the instance of the concept name matches with FileN. The AssignFileN () assigns the FileN number to the instance name.

SearchInstance () searches for instances of concept. The “Class” element is a child element of the “ClassAssertion” element as:

```
“<ClassAssertion>
        <Class IRI="#concept name"/>
        <NamedIndividual IRI="#instance name"/>
</ClassAssertion>” OWL tag.
```

In the case where the IRI attribute of the “Class” element can be found, it returns all IRI attribute values of the “NamedIndividual” element as instance name. All instances of a concept name are used for accessing their property values. Otherwise, it returns false.

CountLocation () counts and assigns the value of the “hasLocation” property of all instances of concept name. Each “hasLocation” value of the instance is counted as an occurrence. The function searches for the IRI attribute of the “DataProperty” element. The “DataProperty” element is a child element of the “DataPropertyAssertion” element as:

```
“<DataPropertyAssertion>
        <DataProperty IRI="property name"/>
        <NamedIndividual IRI="instance name"/>
        <Literal datatypeIRI="&rdf;PlainLiteral">property value</Literal>
</DataPropertyAssertion>” OWL tag.
```

In the case where the “hasLocation” value can be found, the function counts the occurrence of the same property value of each instance location.

SearchSimilar () searches for similar objects from the similar concepts by selecting the objects that appear in the recent FileN. The recent FileN is the latest FileN. It indicates semantic knowledge of the similar physical objects that were located in the recent scan. First, SearchSimilar () searches for parents of the concept. Second, parents of the concept are link concepts. The similar concepts are concept names that match with the link concept from:

```
“<EquivalentClasses>
        <Class IRI="#concept name" />
        <ObjectSomeValuesFrom>
                <ObjectProperty IRI="#HasA" />
                <Class IRI="#link concept" />
        </ObjectSomeValuesFrom>
<EquivalentClasses>” and/or
```

```
“<EquivalentClasses>
        <Class IRI="#concept name" />
        <ObjectSomeValuesFrom>
                <ObjectProperty IRI="#MadeOf" />
                <Class IRI="#link concept" />
        </ObjectSomeValuesFrom>
<EquivalentClasses>” OWL tag.
```

Finally, SearchSimilar () searches for instances of similar objects in the recent FileN. The instance name retrieves from

```
“<DataPropertyAssertion>
        <DataProperty IRI="#hasTime" />
        <NamedIndividual IRI="#instance name" />
        <Literal datatypeIRI="&rdf;PlainLiteral">time value</Literal>
</DataPropertyAssertion>” OWL tag.
```

Example 5.3:

Given owl_document is Figure 5.3 and concept name is cocktail sent from Query. SearchInstance () searches for the “cocktail” instance of the “cocktail” concept in the OWL document. It returns the instance name as “cocktail_1” and “cocktail_2” as:

```
“<ClassAssertion>
      <Class IRI="#cocktail"/>
      <NamedIndividual IRI="#cocktail_1"/>
</ClassAssertion>
<ClassAssertion>
      <Class IRI="#cocktail"/>
      <NamedIndividual IRI="#cocktail_2"/>
</ClassAssertion>” OWL tag.
```

The “IRI” attribute of the “Class” element indicates the concept name from Query. The “IRI” attribute of the “NamedIndividual” element indicates the instance of the concept name.

The SortHasTime () sorts the “hasTime” values of all instances of “cocktail” in the OWL document. The result of FileN number 1 is “20150819113959” and of FileN number 2 is “20150919113959”. The results are received from:

```
“<DataPropertyAssertion>
        <DataProperty IRI="#hasTime"/>
        <NamedIndividual IRI="#cocktail_1"/>
        <Literal datatypeIRI="&rdf;PlainLiteral">20150819113959</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
        <DataProperty IRI="#hasTime"/>
        <NamedIndividual IRI="#cocktail_2"/>
        <Literal datatypeIRI="&rdf;PlainLiteral">20150919113959</Literal>
</DataPropertyAssertion>” OWL tag.
```

The AssignFileN () assigns the number of FileN to the “cocktail_1” and the “cocktail_2” instance. The “cocktail_1” instance appears in FileN number 1 and the “cocktail_2” instance appears in FileN number 2. The CountLocation () function counts the “hasLocation” value of each instance. The “cocktail” instance value has values “1, 0, 1” and “1, 1, 1”. Each “hasLocation” value is counted as an occurrence. The results return the “1, 0, 1” instance location as 1 and the “1, 1, 1” instance location as 1. The results are received from:

```
“<DataPropertyAssertion>
    <DataProperty IRI="#hasLocation"/>
       <NamedIndividual IRI="#cocktail_1"/>
              <Literal datatypeIRI="&rdf; PlainLiteral">1, 0, 1</Literal>
 </DataPropertyAssertion>
<DataPropertyAssertion>
    <DataProperty IRI="#hasLocation"/>
       <NamedIndividual IRI="# cocktail_2"/>
              <Literal datatypeIRI="&rdf; PlainLiteral">1, 1, 1</Literal>
</DataPropertyAssertion>” OWL tag.
```

SearchSimilar () searches for the similar “cocktail” objects by finding the similar concepts that have “HasA” and/or “MadeOf” relations with concept name in Robot Ontology. The “cocktail” concept is a child concept of the “alcohol” concept. The results show the “beer” concept and the “wine” concept. They have “HasA” and/or “MadeOf” relations with the “alcohol” concept.

```
“<EquivalentClasses>
       <Class IRI="#beer" />
              <ObjectSomeValuesFrom>
                     <ObjectProperty IRI="#HasA" />
                     <Class IRI="#alcohol" />
              </ObjectSomeValuesFrom>
</EquivalentClasses>
<EquivalentClasses>
       <Class IRI="#wine" />
              <ObjectSomeValuesFrom>
                     <ObjectProperty IRI="#HasA" />
                     <Class IRI="#alcohol" />
              </ObjectSomeValuesFrom>
<EquivalentClasses>” OWL tag.
```

Next, SearchSimilar () searches for instances of similar concepts in the recent FileN. The recent FileN has “20150919113959” value. In the case where the "beer" instance is found in the recent FileN, the result of similar concept is "beer" that is retrieved from

```
"<DataPropertyAssertion>
        <DataProperty IRI="#hasTime"/>
        <NamedIndividual IRI="#beer_1"/>
        <Literal datatypeIRI="&rdf;PlainLiteral">20150919113959</Literal>
</DataPropertyAssertion>" OWL tag.
```

5.4 Summary

The Semantic Knowledge Acquisition process within the DIRAOF framework is presented through four modules: Ontology Updating, Semantic Representation, NFile and Object Prediction. The studies on the Semantic Knowledge Acquisition process show that OWL tags are updated into Robot Ontology and the Tracking Instances can be calculated. The OWL tags in Robot Ontology indicate concepts, relations between concepts, instances, properties and values. The Tracking Instances show the usage of semantic knowledge to predict the future location of the concept. The Query and the Result Evaluation processes are described in Chapter 6 in order to validate the framework by using the Query and Result evaluation processes.

CHAPTER 6 QUERY, RESULT EVALUATION AND SYSTEM VALIDATION

This chapter provides the detail of the Query and Result Evaluation processes. It also reports the system validation of the DIRAOF framework.

The ontology generated in the Automatic Ontology process using WordNet, ConceptNet and Web Documents data sources needs to be evaluated to ensure the correctness of the concepts, relations and instances before utilising the ontology.

The Query process generates queries to automatically access the hierarchy of the ontology without human involvement. Result Evaluation works with the Query process to assess the correctness of Robot Ontology. The idea to solve the problem of evaluation of the correctness of Robot Ontology is comparing the standard deviation of the semantic similarity value between the pair of concepts in both Robot Ontology and WordNet.

There are two methods for checking the validation of an ontology. First, Local checking is the method to check only the newly created concept. The framework assumes the newly created concept is correct, thus the Local checking is unnecessary because the framework always checks the parent of the newly created concept that exists in WordNet. Second, Global checking is the method to check all concepts in the concept hierarchy in Robot Ontology after ontology components are created by the Automatic Ontology process. The validation process uses Global checking to check the overall structure is still correct after adding newly created concepts.

Global checking should be done after each time that a newly created concept is added to the existing ontology. There are three parameters from the Global checking: standard deviation, semantic similarity and overall score of correctness of the ontology. Section 6.2.2 is used as an example to show the process and the meanings of the three parameters.

The semantic similarity measures how related two concepts are. The range of scores is between 0 and 1. A score that approaches 1 indicates two concepts are closely related. Second, standard deviation measures the dispersion of a set data. A low standard deviation indicates the data is clustered closely around the mean; the data is more reliable. Overall, the standard deviation of the ontology structure indicates how well the ontology is created.

6.1 Query Process and Result Evaluation Process

The information flow is given in Figure 6.1. It shows information exchange between the Query and Result Evaluation processes and those with other related processes. Query automatically searches for the concept name from Robot Ontology and then sends the concept name to Robot Ontology via Result Evaluation. Result Evaluation assesses the correctness of the taxonomy of concepts in Robot Ontology. Moreover, Query sends the concept name to Semantic Knowledge Acquisition to query the semantic knowledge as presented in Chapter 5. Robot Ontology is accessed and is evaluated during this chapter.

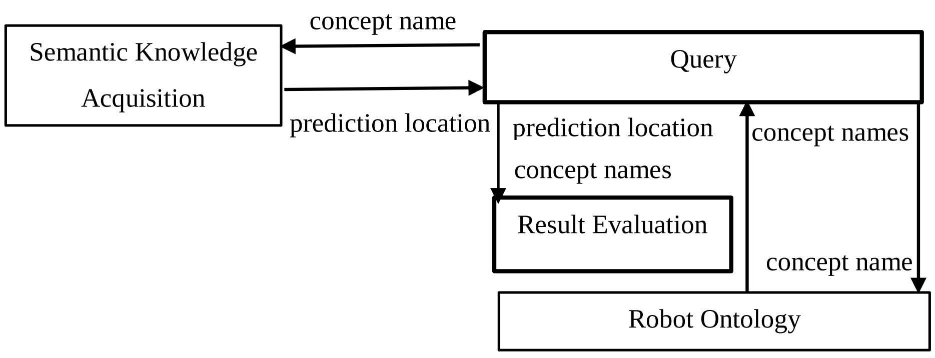


Figure 6.1 Information flow between Query and Result Evaluation

The step-by-step process of information exchange is as follows:

- The Query process randomly chooses a concept name from Robot Ontology as a query and sends the query to Semantic Knowledge Acquisition.
- The Semantic Knowledge Acquisition process searches for the instances of the concept and returns the location of the concept.

- The Query process searches for child concepts of the concept name.
- The Query process sends the concept name, child concept name(s) and the prediction location to the Result Evaluation process.
- The Result Evaluation process calculates the semantic similarity between the concept name and child concept name(s) and calculates the standard deviation.
- The Result Evaluation process stores and the prediction location, the semantic similarity score and the standard deviation value as a text file.

6.1.1 Query Process

The Query is a process of:

- choosing a concept name from Robot Ontology as a query and sending the concept name to Semantic Knowledge Acquisition
- searching for child concepts of the concept name from Robot Ontology
- sending the concept name and child concept name(s) to the Result Evaluation process.

The Query process selects a concept name from Robot Ontology in order to query the prediction location of the concept from Semantic Knowledge Acquisition. The Query process searches for child concepts of the concept name from Robot Ontology and sends the concept name and child concept name(s) to the Result Evaluation process to evaluate the structure of the concept name and child concept name(s).

The QueryConcept algorithm is given in Figure 6.2, PL represents prediction location, C stands for concept, SC represents sibling concepts, SKA stands for Semantic Knowledge Acquisition and CH stands for child concepts. The concept is sent to Semantic Knowledge Acquisition, and Semantic Knowledge Acquisition sends the prediction location back as the input of the algorithm. The loop is to choose the concept in WordNet. A loop with the While condition (Last C) searches for child concepts of the concept until the last concept in Robot Ontology. In the case that the concept is a leaf node, the algorithm searches for sibling concepts, assigns the sibling concept as child concept and assigns parent of the concept as concept.

```
Algorithm: QueryConcept
Input:      PL
Begin
        C ←QueryConcept ();
        Send C to SKA and retrieve PL as input;
        Do {
                If C is leaf node then
                        SC ← SearchSiblings (C);
                        CH←SC;
                        C←parents of C;
                Else
                        CH←SearchChilds(C);
        } While (Last C)
Return C, CH, PL;
```

Figure 6.2 QueryConcept algorithm

This step uses the following three functions:

- QueryConcept () – choosing the concept in Robot Ontology
- SearchSiblings () – searching for sibling concepts
- SearchChilds () – searching for child concepts.

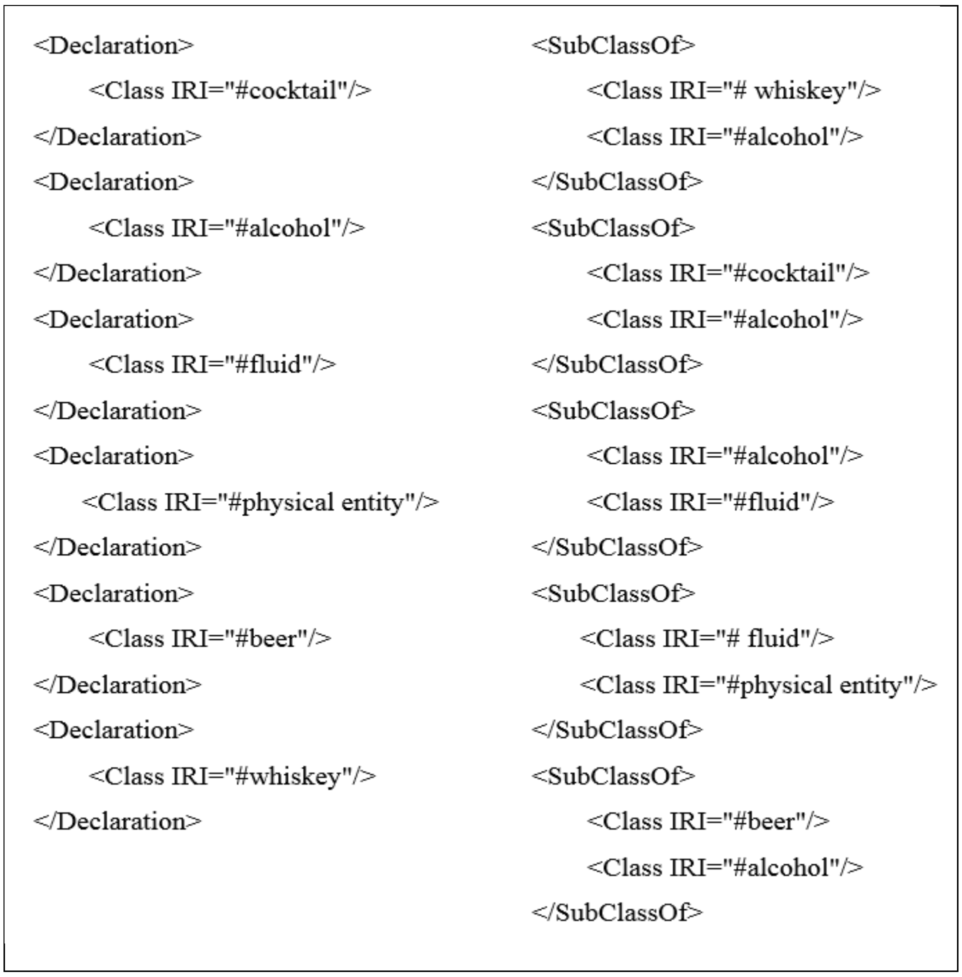
QueryConcept searches for and selects concept names in Robot Ontology.
SearchSiblings searches for sibling concepts. First, the function searches for parent concepts of the concept name. Second, the function searches for child concepts of parent concepts and assigns them as sibling concepts.
SearchChilds searches for child concepts. The function searches for child concepts of the concept.

Example 6.1:

The algorithm in Figure 6.2 calls the QueryConcept () function to select a concept name in Robot Ontology by searching for the “IRI” attribute of the “Class” element that is a child element of the “Declaration” element. The “cocktail” concept is selected. The algorithm sends the “cocktail” concept to Semantic Knowledge Acquisition and Semantic Knowledge Acquisition sends the prediction location of the “cocktail” concept back to Query process.

The “cocktail” concept is a leaf node that does not have child nodes; then the algorithm calls the SearchSiblings () function to search for parent concepts of the “cocktail” concept. The parent concept of the “cocktail” concept is the “alcohol” concept. After that, SearchSiblings () searches for child concepts of the “alcohol” concept. The results are the “whiskey” and “beer” concepts as the “SubClassOf” element of the “alcohol” concept in Figure 6.3.

The “alcohol” concept (parent of the “cocktail” concept) is assigned as the concept. The “whiskey” and “beer” concepts (child concepts of the “alcohol” concept) are assigned as child concepts.



```
<Declaration>
      <Class IRI="#cocktail"/>
</Declaration>
<Declaration>
      <Class IRI="#alcohol"/>
</Declaration>
<Declaration>
      <Class IRI="#fluid"/>
</Declaration>
<Declaration>
     <Class IRI="#physical entity"/>
</Declaration>
<Declaration>
      <Class IRI="#beer"/>
</Declaration>
<Declaration>
      <Class IRI="#whiskey"/>
</Declaration>
```

```
<SubClassOf>
      <Class IRI="# whiskey"/>
      <Class IRI="#alcohol"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#cocktail"/>
      <Class IRI="#alcohol"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#alcohol"/>
      <Class IRI="#fluid"/>
</SubClassOf>
<SubClassOf>
     <Class IRI="# fluid"/>
     <Class IRI="#physical entity"/>
</SubClassOf>
<SubClassOf>
      <Class IRI="#beer"/>
      <Class IRI="#alcohol"/>
</SubClassOf>
```

Figure 6.3 Part of OWL tags in Robot Ontology

6.1.2 Result Evaluation Process

Result Evaluation applies the Global checking method to check the overall structure of Robot Ontology after adding newly created concepts in concept hierarchy. Result Evaluation retrieves a concept name, its child concept(s) and the prediction location of the concept from the Query process. Result Evaluation assesses the concept and its child concept(s). Robot Ontology contains an "is-a" hierarchy between the parent concept and child concepts. The following operations are presented below in order to show all child concepts are related to a concept name.

- calculating the semantic similarity of a pair of a concept name and child concept(s)
- calculating the standard deviation of the semantic similarity value and
- keeping the prediction location of the concept, the semantic similarity score and the standard deviation value as a text file.

Due to the structure based or edge counting semantic measure being based on "is-a" hierarchy links between concepts, it computes the semantic similarity measure in the hierarchy of the ontology. The length of the path linking the concepts and the position of the concepts in the taxonomy are counted.

```
Algorithm: ResultEvaluation
Input:      C, CH, PL
Begin
    StoreResults (C, PL);
    Do {
        SS← WordSimilarity (C, CH);
        StoreResults (C, CH, SS);
    } While (Last CH of C)
    SD← StandardDeviation (SS);
    StoreResults (C, SD);
```

Figure 6.4 ResultEvaluation algorithm

The ResultEvaluation algorithm is given in Figure 6.4. In this algorithm, C stands for concept, CH represents child concepts, PL represents prediction location, SD stands for standard deviation and SS stands for semantic similarity score. First, the algorithm stores the prediction location of the concept. The condition of the While loop (Last CH of C) calculates the semantic similarity between the concept and child concepts. It processes until the last child concept of the concept. After that, the algorithm calculates the standard deviation and stores the result. This algorithm uses the following three functions:

- StoreResults () – keeping the prediction location of the concept as a text file
- WordSimilarity () – calculating semantic similarity score for each concept and its child concepts
- StandardDeviation () – calculating the standard deviation of the semantic similarity of the concept.

StoreResults keeps the results as a text file. The function connects texts together in StoreResult.txt. The pattern of text is name, bookmark and value. The name is a name of a concept. The bookmark indicates the type of result. There are three types of results. The "@" bookmark indicates location of an instance from Semantic Knowledge Acquisition. In the case of the semantic similarity score between the concept and child concepts, the texts are concept name, "#" bookmark and child concept name. The "#" bookmark indicates the semantic similarity score between the concept and child concepts. The "$" bookmark indicates the standard deviation value of the concept name.

Semantic similarity measures between two concepts in hierarchy structure. It is calculated from Wu and Palmer (1994). This method is suitable for computing the semantic similarity in an ontology hierarchy. A high semantic similarity score of two concepts means the two concepts are the closely related. A low semantic similarity score indicates a lack of cohesion of the two concepts. The semantic similarity measure score is between 0 and 1.

The standard deviation is a quantity measure in statistics. It measures the dispersion of a set of data. A low standard deviation of the semantic similarity between parent concept and child concepts indicates the child concepts relate to the parent concept. It means that the hierarchy of the parent concept and child concepts in Robot Ontology is correct. On the other hand, a high standard deviation means the data spread over the mean of the set of data. It means that the hierarchy in Robot Ontology tends to be incorrect.

Example 6.2:

WordSimilarity () calculates the semantic similarity score between “alcohol” and “cocktail”, between “alcohol” and “whiskey” and between “alcohol” and “beer” concepts. StoreResults () stores concept, prediction location, standard deviation, semantic similarity score between “cocktail” and “whiskey” and between “cocktail” and “beer” concepts. The semantic similarity between “alcohol” and “cocktail”, between “alcohol” and “whiskey” and between “alcohol” and “beer” is 0.88. Standard deviation is 0.

The text file stores: “cocktail@1,0,0alcohol#cocktail#0.8 alcohol#whiskey#0.8 alcohol#beer#0.8alcohol$0”.

6.2 System Validation

Chapters 3 to 5 presented the components of the DIRAOF framework for creating a dynamic ontology, namely, Automatic Ontology and Semantic Knowledge Acquisition. The system validation aims to show that the DIRAOF framework is able to learn and build a dynamic ontology. The expected results from the DIRAOF framework are the correct concepts and relations between concepts in concept hierarchy.

The semantic similarity and standard deviation are criteria for evaluating the concept hierarchy in Robot Ontology. In the case that the semantic similarity between the two concepts meets a minimum word-to-word threshold of 0.5 (Lintean, 2012), the two concepts are related. The threshold is fixed to accept the similarity between two concepts. In the case that the standard deviation of each concept is nearly 0, the semantic similarity score is clustered closely around the mean. A concept hierarchy that has a low standard deviation is more reliable than a concept hierarchy that has a high standard deviation value.

6.2.1 Setting

The ontology is empty at the beginning. There are 200 object names in the household environment as given in Table 6.1. Object names are categorised into 9 categories. The category names are pastry, drink, food, fruit and vegetable, meat and fish, perishable, product, medicine and kitchenware. The category names were suggested by various sources from internet.

Table 6.2 shows the property values associated with the object names given in Table 6.1. As mentioned earlier, the “date and time” and “location” properties are mandatory. The framework can randomly choose object names from Table 6.1 and their property values from Table 6.2. The object names and property values form a text file in which the number of object names can vary. Each file consists of a unique time value and each line in a file consists of a location value of each object name, other property name(s) and the object name. For example, the first file consists of four object names and all object names have “20150819113959” as the time value. The “cocktail” object name has “1,1,1” as the location value and “frozen” as the property value. The “beer” object name has “0,0,1” as the location value. The “milk” object name has “0, 3, 0” as the location value and “semi-skimmed” as the property value. The “cheese” has “1, 1, 0” as the location value.

To test the Semantic Knowledge Acquisition process, 10 text files were generated in this way, as given in Figure 6.5.

The framework is linked to WordNet, ConceptNet and Web Documents in order to create concepts and relations between concepts. The WordNet 3.1 database is stored in ASCII format. The ConceptNet 5 Web API for accessing is available from http://conceptnet5.media.mit.edu/data/5.4/search. Web Documents searches are for category names of object names that do not exist in WordNet.

6.2.2 Testing and Results

Concept Creation, Relation Creation and OWL Creation in the Automatic Ontology process receive object names from the text files in order to create concepts and relations between concepts. Data and Information Retrieval provides concept names for Concept Creation and provides concept names and relations between concept names for Relation Creation. It recognises concepts, hierarchy and non-hierarchy relations with WordNet, ConceptNet and Web Documents. Instance Creation creates instances of concepts and Property Creation receives property values to classify property names from property values.

Referring to Section 4.2, Concept Creation assumes the object name as the concept name in WordNet. In the case where the concept name's parent is in WordNet and the concept name's antecedent is the "physical entity" concept, the ConceptCreation algorithm accepts the object name as a concept and creates the concept name in Robot Ontology. Using the first text file as an example, the process of Automatic Ontology can be explained as follows.

Concept Creation assumes a "cocktail" object name as a "cocktail" concept name in WordNet. In the case where the "cocktail" concept name's parent ("course" and "alcohol" concepts) is in WordNet and the "cocktail" concept name's antecedent is the "physical entity" concept, the ConceptCreation algorithm accepts a "cocktail" object name as a concept and creates the "cocktail" concept name in Robot Ontology and the concept name is sent to Relation Creation in order to create concepts and relations between concepts. The result of the RelationCreation algorithm is the text file that presents the relations between the concepts. The concept_relation_text is an "is-a" relation between "cocktail" and "course", "course" and "food", "food" and "matter", "matter" and "physical entity", "cocktail" and "alcohol", "alcohol" and "food", "alcohol" and "drug", "alcohol" and "fluid", "drug" and "physical entity", "fluid" and "physical entity", as given in Table 6.3. The "HasA" and "MadeOf" relations are not created due to the first concept in Robot Ontology. OWL Creation generates OWL tags from concept relation text that are obtained from Relation Creation.

The first file

```
20150819113959
1,1,1 frozen cocktail
0,0,1 beer
0,3,0 semi-skimmed milk
1,1,0 cheese
```

The second file

```
20150919113959
1,0,0 jam
1,1,0 cheese
-1,1,1 slow-cooked minted gravy
1,0,1 biscuit
0,1,1 sweet salty popcorn
```

The third file

```
20151019113959
1,1,1 konjac
0,0,1 pepsi
0,3,0 semi-skimmed milk
```

The forth file

```
20151119113959
-1,1,1 minted gravy
1,0,1 biscuit
0,1,1 sweet salty popcorn
```

The fifth file

```
20151120113959
0,3,0 semi-skimmed milk
```

The sixth file

```
20151125113959
1,1,1 konjac
0,0,1 semi-skimmed milk
0,3,0 pepsi
-1,1,1 biscuit
1,0,1 gravy
```

The seventh file

```
20151130113959
1,1,1 pepsi
0,0,1 milk
0,3,0 konjac
```

The eighth file

```
20151225113959
1,1,1 pepsi
0,0,1 milk
0,3,0 konjac
-1,1,1 biscuit
1,0,1 gravy
```

The nineth file

```
20151230113959
1,0,1 gravy
1,0,0 milk
```

The tenth file

```
20151231113959
1,1,0 cheese
```

Figure 6.5 The example input files in the experiment

Table 6.1 Example object names in nine category names

pastry	drink	food	fruit and vegetable	meat and fish	perishable	product	medicine	kitchenware	
loaf	pepsi	ketchup	orange	cod	cheddar	glove	cough syrup	storage	chopper
baguette	coke	mayonnaise	avocado	pork	cheese	shampoo	pastilles	jug	blender
biscuit	lemonade	gravy	spinach	chop	egg	soap	syrup	peeler	grinder
bread	kronenbourg	popcorn	potatoes	mince	butter	lotion	plasters	poacher	steamer
mochi	orangeade	mustard	broccoli	prawn	jam	spray	bandage	opener	jar
daifuku	juice	rice	carrot	fishcakes	pastry	dishwasher	ointment	spoon	fryer
ice cream cake	water	sugar	cauliflower	mussel	buttermilk	tissue	lozenges	baking tray	scoop
flaugnarde	squash	noodle	celery	salmon	margarine	liners	inhalator	funnel	mortar
	cordial	cereal	tomato	steak	mascarpone	rack	nicorette	fork	nutribullet
	ale	tagliatelle	mushroom	sea bass	quiche	brush	senokot	spatula	sandwich press
	beer	chow mein	pear	oyster	pizza	toothbrush	windsetlers	mug	corkscrew
	cocktail	linguine	garlic	fowl	coleslaw	roller	revitalens	dish	
	punch	spaghetti	beans	goose	smoothies	duster	germolene	plate	
	milk	pasta	grapefruit	lamb	juice	plunger	nicolites cartomiser menthol	bowl	
	champagne	instant noodle	melon	squid	clementine	toilet roll	decongestant	shaker	
	chocolate	remoulade	mango		tteokbokki	bin	linctus	teaspoon	
	tea	ramen	lime		chikuwa	conditioner		saucepan	
	coffee	miso	lemon		okonomiyaki	cigars		pan	
	strawberry daiquiri	sauerkraut	banana		nabemono	lighter		glass	
	cocoa	basmati rice	nectarine		pfefferpotthast	candle		tumbler	
	malt		parsnip		spanferkel	bulb		coasters	
	lager		ginger		fajita	toothpaste		muffin tin	
	cider		grape					hand mixer	
	bitter		raspberries					cake tester	
	chardonnay		cucumber					cafetiere	
	merlot		strawberry					saltball	
	prosecco		konjac					pot	
	sauvignon							cooker	
	jameson								
	mojito								
	yakult								

The result of the algorithm consists of concepts and relations between concepts as given in Figure 6.6. Instance Creation creates the “cocktail_1” instance of the “cocktail” concept as:

```
“<Declaration>
        <NamedIndividual IRI="#cocktail_1"/>
</Declaration>
<ClassAssertion>
        <Class IRI="#cocktail"/>
        <NamedIndividual IRI="#cocktail_1"/>
</ClassAssertion>” OWL tags in Robot Ontology.
```

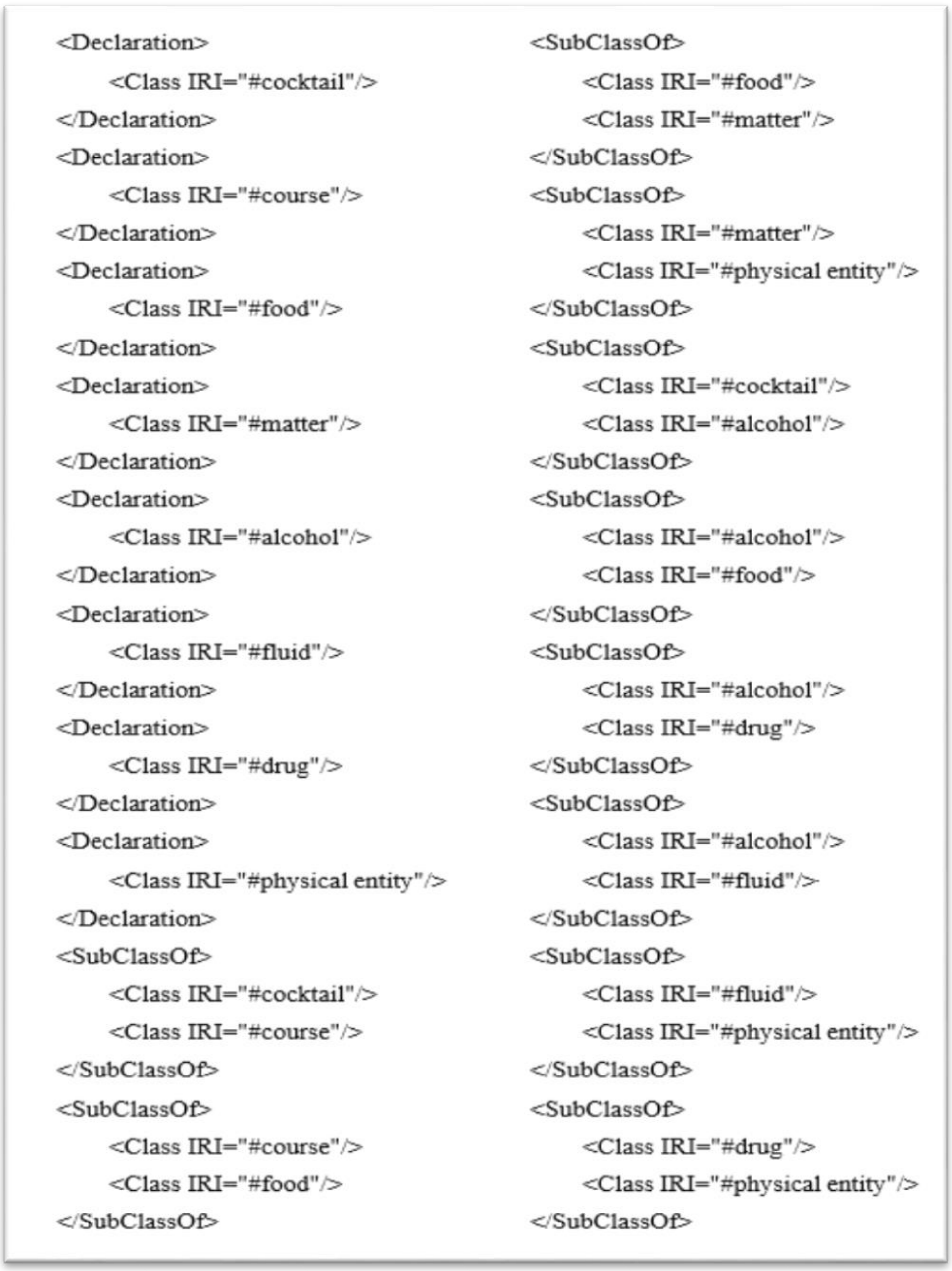
The concepts, relations between concepts and instances of the “cocktail” concept are sent to Robot Ontology.

Table 6.2 Examples of property values of object names

Adjectives		Object names	Category names
1,1,1		biscuit	bakery
0,0,1		pepsi	drinks
0,1,1		beer	drinks
1,0,0,	frozen	cocktail	drinks
1,0,1,	semi-skimmed	milk	drinks
0,1,0,	minted, slow-cooked	gravy	food
0,-1,0,	sweet, salty	popcorn	food
0,1,-1		konjac	fruit and vegetable
1,-1,1		cheese	perishable
1,1,-1		jam	perishable

Table 6.3 The output of the “cocktail” object name and the “frozen” value

Concept and Relations				Semantic Knowledge		
is-a		HasA	MadeOf	instance name	property name	value
cocktail	course			cocktail_1	hasTime	20150819113959
course	food				hasLocation	1,1,1
food	matter				hasColour	
matter	physical entity				hasShape	
cocktail	alcohol				hasProperty	frozen
alcohol	food					
alcohol	drug					
alcohol	fluid					
drug	physical entity					
fluid	physical entity					



```
<Declaration>
    <Class IRI="#cocktail"/>
</Declaration>
<Declaration>
    <Class IRI="#course"/>
</Declaration>
<Declaration>
    <Class IRI="#food"/>
</Declaration>
<Declaration>
    <Class IRI="#matter"/>
</Declaration>
<Declaration>
    <Class IRI="#alcohol"/>
</Declaration>
<Declaration>
    <Class IRI="#fluid"/>
</Declaration>
<Declaration>
    <Class IRI="#drug"/>
</Declaration>
<Declaration>
    <Class IRI="#physical entity"/>
</Declaration>
<SubClassOf>
    <Class IRI="#cocktail"/>
    <Class IRI="#course"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#course"/>
    <Class IRI="#food"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#food"/>
    <Class IRI="#matter"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#matter"/>
    <Class IRI="#physical entity"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#cocktail"/>
    <Class IRI="#alcohol"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#alcohol"/>
    <Class IRI="#food"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#alcohol"/>
    <Class IRI="#drug"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#alcohol"/>
    <Class IRI="#fluid"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#fluid"/>
    <Class IRI="#physical entity"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#drug"/>
    <Class IRI="#physical entity"/>
</SubClassOf>
```

Figure 6.6 OWL tags of the “cocktail” concept

Property Creation classifies the property name from the property values and sends OWL tags as a text file as given in Figure 6.7 to the Semantic Knowledge Acquisition process.

```
<DataPropertyAssertion>
        <DataProperty IRI="#hasTime"/>
        <NamedIndividual IRI="#cocktail_1"/>
        <Literal datatypeIRI="&rdf;PlainLiteral">20150819113959</Literal>
</DataPropertyAssertion>
<DataPropertyAssertion>
       <DataProperty IRI="#hasLocation"/>
            <NamedIndividual IRI="#cocktail_1"/>
                        <Literal datatypeIRI="&rdf;PlainLiteral">1, 1, 1</Literal>
 </DataPropertyAssertion>
<DataPropertyAssertion>
       <DataProperty IRI="#hasProperty"/>
            <NamedIndividual IRI="#cocktail_1"/>
                        <Literal datatypeIRI="&rdf;PlainLiteral">frozen</Literal>
 </DataPropertyAssertion>
```

Figure 6.7 OWL tags of the “cocktail_1” instance

Semantic Knowledge Acquisition associates semantic knowledge of the “cocktail” instance with the “cocktail” instance in Robot Ontology. The “cocktail” instance has a “hasTime” property name with “20150819113959” value, a “hasLocation” name with “1,1,1” value and a “hasProperty” property name with “frozen” value. The next object names, the “beer” object name with “0,0,1” value, “milk” object name with “0,3,0” and “semi-skimmed” value, “cheese” object name with “1,1,0” value in the first input file, are processed and the results are as given in Table 6.4.

Table 6.4 The results of the first input file

object name	value	Concepts and Relations			
		is-a		HasA	MadeOf
cocktail	1,1,1	cocktail	course		
	frozen	course	food		
		food	matter		
		matter	physical_entity		
		cocktail	alcohol		
		alcohol	food		
		alcohol	drug		
		alcohol	fluid		
		drug	physical_entity		
		fluid	physical_entity		
beer	0,0,1	beer	alcohol	alcohol	
milk	0,3,0	milk	food		
	semi-skimmed	milk	stream		
		stream	thing		
		stream	physical_entity		
		thing	physical_entity		
		milk	liquid		
		milk	body_substance		
		body_substance	physical_entity		
cheese	1,1,0	cheese	solid		milk
		cheese	foodstuff		
		solid	physical_entity		
		foodstuff	physical_entity		

Semantic Knowledge		
instance name	property name	value
cocktail_1	hasTime	20150819113959
	hasLocation	1,1,1
	hasProperty	frozen
beer_1	hasTime	20150819113959
	hasLocation	0,0,1
milk_1	hasTime	20150819113959
	hasLocation	0,3,0
	hasProperty	semi-skimmed
cheese_1	hasTime	20150819113959
	hasLocation	1,1,0

After creating the “cocktail” concept, ConceptCreation creates the “beer” concept as a child concept of the “alcohol” concept. The ConceptCreation algorithm creates a “HasA” relation between the “beer” and “alcohol” concepts by using ConceptNet, as the “alcohol” concept already exists in Robot Ontology. The “cheese” concept also has a “MadeOf” relation with the “milk” concept due to the “milk” concept already existing in Robot Ontology. ConceptNet is used for building “HasA” and “MadeOf” relations between physical entity concepts.

Results from the ten input files are present in Table 6.5. They consist of concepts and relations between concepts. There are 39 concepts and 59 relations from the ten input files. After concepts, relations between concepts and instances are created. Semantic Knowledge Acquisition keeps a record of the semantic knowledge of the instances as given in Table 6.6 in order to locate objects in the dynamic environment. There are 32 instances in Robot Ontology. The “cocktail”, “beer” and concepts have one instance (cocktail_1, beer_1). The “popcorn” concept has two instances. The “cheese” concept has three instances. The “biscuit”, “corm” and “pepsi” concepts have four instances. The “gravy” concept has five instances. The “milk” concept has seven instances.

The Query process retrieves 39 concept names from Robot Ontology and sends each concept name to Semantic Knowledge Acquisition. Semantic Knowledge Acquisition locates instances of the concepts’ location by Tracking Instances for predicting the future location. For example, the “milk” concept is sent from Query to Semantic Representation in Semantic Knowledge Acquisition. Decision scores are calculated by equation (5.3) in Section 5.4 for each location of the “milk” instance. The “milk” instances are found at FileN as 1,3,5 and at “0,3,0” location at 3 times, as 6,7,8 at “0,0,1” location at 3 times, and as 9 at “1,0,0” location at 1 time. The decision score of each location can be calculated as given in Table 6.7. The “1,0,0” location is selected as the future location of the “milk” instance due to it having the highest decision score (0.55).

The number of concepts, relations between concepts and instances of concepts depend on the physical object names and property values in the household environment. The object names in Table 6.1 are processed randomly. The experiments assume that all 200 object names are processed as given from Table 6.8 to Table 6.16. Table 6.8 to Table 6.16 show the results of pastry, drink, food, fruit and vegetable, meat and fish, perishable, product, medicine and kitchenware, respectively. The structure of each table consists of object name, concept name and parents of concept name. The “is-a”, “HasA” and “MadeOf” relation between concept names and parents of concept name are present in appendix A.

Table 6.5 Concepts and relations between concepts from ten input files

object name	value	Concepts and Relations			
		is-a		HasA	MadeOf
cocktail	1,1,1	cocktail	course		
	frozen	course	food		
		food	matter		
		matter	physical_entity		
		cocktail	alcohol		
		alcohol	food		
		alcohol	drug		
		alcohol	fluid		
		drug	physical_entity		
		fluid	physical_entity		
beer	0,0,1	beer	alcohol	alcohol	
milk	0,3,0	milk	food		
	semi-skimmed	milk	stream		
		stream	thing		
		stream	physical_entity		
		thing	physical_entity		
		milk	liquid		
		milk	body_substance		
		body_substance	physical_entity		
cheese	1,1,0	cheese	solid		milk
		cheese	foodstuff		
		solid	physical_entity		
		foodstuff	physical_entity		
jam	1,0,0	jam	confiture		
		confiture	dainty		
		dainty	food		
		jam	gathering		
		gatering	event		
		gatering	group		
		event	physical_entity		
		group	physical_entity		
gravy	-1,1,1	gravy	condiment		
	minted	condiment	ingredient		
	slow-cooked	ingredient	food		
		gravy	event		
		event	physical_entity		
		gravy	foodstuff		
biscuit	1,0,1	biscuit	baked_goods		
		baked_goods	solid		
		biscuit	bread		
		bread	food		
popcorn	0,1,1	popcorn	grain		
	sweet	grain	grass		
	salty	grass	herb		
		herb	substance		
		herb	physical_entity		
		grass	physical_entity		
		grain	food		
		popcorn	cereal		
		cereal	food		
		substance	physical_entity		
konjac		corm	plant_organ		
		plant_organ	physical_entity		
pepsi	0,0,1	pepsi	soft_drink		
		soft_drink	liquid		
		liquid	matter		
		soft_drink	food		

Table 6.6 Semantic knowledge of ten input files

Semantic Knowledge		
instance name	property name	value
cocktail_1	hasTime	20150819113959
	hasLocation	1,1,1
	hasProperty	frozen
beer_1	hasTime	20150819113959
	hasLocation	0,0,1
milk_1	hasTime	20150819113959
	hasLocation	0,3,0
	hasProperty	semi-skimmed
cheese_1	hasTime	20150819113959
	hasLocation	1,1,0
jam_1	hasTime	20150919113959
	hasLocation	5,0,0
cheese_2	hasTime	20150919113959
	hasLocation	1,1,0
gravy_1	hasTime	20150919113959
	hasLocation	-1,1,1
	hasProperty	minted
	hasProperty	slow-cooked
biscuit_1	hasTime	20150919113959
	hasLocation	1,0,1
popcorn_1	hasTime	20150919113959
	hasLocation	0,1,1
	hasProperty	sweet
	hasProperty	salty
corm_1	hasTime	20151019113959
	hasLocation	10,0,0
pepsi_1	hasTime	20151019113959
	hasLocation	0,0,1
milk_2	hasTime	20151019113959
	hasLocation	0,3,0
gravy_2	hasTime	20151119113959
	hasLocation	-1,1,1
biscuit_2	hasTime	20151119113959
	hasLocation	1,0,1
popcorn_2	hasTime	20151119113959
	hasLocation	0,1,1
	hasProperty	sweet
	hasProperty	salty
milk_3	hasTime	20151120113959
	hasLocation	0,3,0
	hasProperty	semi-skimmed
corm_2	hasTime	20151125113959
	hasLocation	1,1,1
milk_4	hasTime	20151125113959
	hasLocation	0,0,1
	hasProperty	semi-skimmed
pepsi_2	hasTime	20151125113959
	hasLocation	0,3,0
biscuit_3	hasTime	20151125113959
	hasLocation	-1,1,1
gravy_3	hasTime	20151125113959
	hasLocation	1,0,1
pepsi_3	hasTime	20151130113959
	hasLocation	1,1,1
milk_5	hasTime	20151130113959
	hasLocation	0,0,1
corm_3	hasTime	20151130113959
	hasLocation	0,3,0
pepsi_4	hasTime	20151225113959
	hasLocation	1,1,1
milk_6	hasTime	20151225113959
	hasLocation	0,0,1
corm_4	hasTime	20151225113959
	hasLocation	0,3,0
biscuit_4	hasTime	20151225113959
	hasLocation	-1,1,1
gravy_4	hasTime	20151225113959
	hasLocation	1,0,1
gravy_5	hasTime	20151230113959
	hasLocation	1,0,1
milk_7	hasTime	20151230113959
	hasLocation	1,0,0
cheese_3	hasTime	20151231113959
	hasLocation	1,1,0

Table 6.7 The decision score of three locations

Location	FileN	NFileN-RFileN	F_{loc}/NFileN	$0.5^{(NFileN-RFileN)}$	RFileN/NFileN	Decision score
0,3,0	1,3,5	9,7,5	0.3	0.03125	0.5	0.315625
0,0,1	6,7,8	4,3,2	0.3	0.25	0.8	0.5
1,0,0	9	1	0.1	0.5	0.9	0.55

Table 6.8 The pastry category

Object name	Concept name	Parents of concept name
loaf	loaf	solid, baked_good
baguette	baguette	white_bread
biscuit	biscuit	baked_good,bread
bread	bread	food,foodstuff
mochi	rice	grass,foodstuff,writer
daifuku	strawberry	edible_fruit,vascular_plant
ice cream cake	cream	foodstuff
flaugnarde	strawberry	edible_fruit,vascular_plant

Table 6.9 The drink category

Object name	Concept name	Parents of concept name
pepsi	pepsi	soft_drink
coke	coke	substance
lemonade	lemonade	beverage
cherryade	cherryade	container, object
orangeade	orangeade	beverage
juice	juice	food, electrical_phenomenon, body_substance
water	water	fluid
squash	squash	athletic_game, produce
cordial	cordial	drug_of_abuse,beverage,
ale	ale	brew
beer	beer	alcohol
cocktail	cocktail	alcohol,course
punch	punch	stroke,alcohol,implement
milk	milk	food, stream, liquid,body_substance
champagne	champagne	wine,geographical_area
chocolate	chocolate	food,solid,liquid
tea	tea	nutriment,food,liquid,party,flavorer
coffee	coffee	food,liquid
strawberry daiquiri	strawberry daiquiri	cocktail
cocoa	cocoa	liquid,food
malt	malt	foodstuff, drink, malt
lager	lager	military_quarters,brew
cider	cider	liquid,food
bitter	bitter	property,beer
chardonnay	chardonnay	wine
merlot	merlot	wine
prosecco	wine	red,drug_of_use,beverage
sauvignon	wine	red,drug_of_use,beverage
jameson	whiskey	alcohol
mojito	juice	food, electrical_phenomenon, body_substance
yakult	container	artifact

Table 6.10 The food category

Object name	Concept name	Parents of concept name
ketchup	ketchup	flavorer
mayonnaise	mayonnaise	sauce
gravy	gravy	condiment, event, foodstuff
popcorn	popcorn	grain,cereal
mustard	mustard	vegetable, herb
rice	rice	grass,foodstuff,writer
sugar	sugar	organic_compound, flavorer, molecule
noodle	noodle	food,head
cereal	cereal	food
tagliatelle	tagliatelle	food
chow mein	chow mein	nutriment
linguine	linguine	food
spaghetti	spaghetti	dish,food
pasta	pasta	nutriment,solid
instant noodle	noodle	food,head
remoulade	shrimp	person,food
ramen	bowl	artifact
miso	condiment	ingredient
sauerkraut	nutriment	substance,physical entity
basmati	container	artifact

Table 6.11 The fruit and vegetable category

Object name	Concept name	Parents of concept name
orange	orange	stream,color,coloring_material
avocado	avocado	produce
spinach	spinach	vegetable,herb
potatoes	potatoes	vegetable,foodstuff
broccoli	broccoli	herb
carrot	carrot	plant_organ,vegetable
cauliflower	cauliflower	herb
celery	celery	produce
tomato	tomato	vascular_plant
mushroom	mushroom	fungus, produce, physical_phenomenon
pear	pear	produce
garlic	garlic	ingredient
beans	beans	herb
grapefruit	grapefruit	edible_fruit
melon	melon	produce
mango	mango	produce
lime	lime	material
lemon	lemon	whole
banana	banana	produce
nectarine	nectarine	produce
parsnip	parsnip	plant_organ,vegetable
ginger	ginger	ingredient,vascular_plant
grape	grape	produce
raspberries	raspberries	edible_fruit
cucumber	cucumber	produce
strawberry	strawberry	edible_fruit,vascular_plant
konjac	corm	plant_organ

Table 6.12 The meat and fish category

Object name	Concept name	Parents of concept name
cod	cod	sheath,seafood
pork	pork	food
chop	chop	meat,natural_phenomenon
mince	mince	food
prawn	prawn	food
fishcakes	fishcakes	region, causal_agent
mussel	mussel	seafood
salmon	salmon	fish,stream,color,food
steak	steak	meat
sea bass	sea bass	fish,seafood
oyster	oyster	seafood
fowl	fowl	food
goose	goose	simpleton,bird,
lamb	lamb	food,person
squid	squid	food

Table 6.13 The perishable category

Object name	Concept name	Parents of concept name
cheddar	cheddar	dairy_product, food,settlement,
cheese	cheese	solid, foodstuff
egg	egg	food
butter	butter	person, solid, foodstuff
jam	jam	confiture, gathering
pastry	pastry	concoction, food
buttermilk	buttermilk	fluid, part
margarine	margarine	condiment
mascarpone	mascarpone	cheese
quiche	quiche	amerindian
pizza	pizza	nutriment
coleslaw	coleslaw	dish
smoothies	smoothies	deceiver
juice	juice	food,electrical_phenomenon,body_substance
clementine	clementine	citrus
tteokbokki	rice	grass,foodstuff,writer
chikuwa	skin	artifact, causal_agent
okonomiyaki	hiroshima	municipality, geographic_point
nabemono	article	whole
pfefferpotthast	span	digit,artifact
spanferkel	pig	unpleasant_person, block, lawman, container
fajita	container	artifact

Table 6.14 The product category

Object name	Concept name	Parents of concept name
glove	glove	clothing
shampoo	shampoo	formulation
soap	soap	club_drug, formulation
lotion	lotion	matter
spray	spray	chemical, vapor, decoration, container, discharge
dishwasher	dishwasher	workman
tissue	tissue	material,part
liners	liners	piece
rack	rack	locomotion, meat, ending, supporting_structure, device
brush	brush	process
toothbrush	toothbrush	implement
roller	roller	bird, solid, pigeon, movement,mechanism, machine
duster	duster	whole
plunger	plunger	person
dishbrush	dishbrush	ester
bleach	bleach	causal_agent,physical_entity
conditioner	conditioner	chemical
cigars	cigars	tobacco
lighter	lighter	substance
candle	candle	light_unit, source_of_illumination
bulb	bulb	body_part, object
toothpaste	toothpaste	cleansing_agent

Table 6.15 The medicine category

Object name	Concept name	Parents of concept name
cough syrup	cough	evidence
pastilles	pastilles	candy
syrup	syrup	flavorer
plaster	plaster	artifact, building_material, substance
bandage	bandage	cloth_covering
ointment	ointment	matter
lozenge	lozenge	medicine, sweet
inhalator	inhalator	container, device
nicorette	auto	self_propelled_vehicle
senokot	senna	woody_plant
windsetlers	storage	facility, possession, device, component
revitalens	lense	device
germolene	state	district, unit
nicolites cartomiser menthol	cigarette	tobacco
decongestant	decongestant	drug
linctus	medicine	agent

Table 6.16 The kitchenware category

Object name	**Concept name**	**Parents of concept name**
storage	storage	facility, possession, device, component
jug	jug	vessel
peeler	peeler	person, entertainer
poacher	poacher	acquirer, container, kitchen_utensil
opener	opener	causal_agent,physical_entity
spoon	spoon	tableware
baking tray	baking tray	kitchen_utensil
funnel	funnel	chimney, implement
fork	fork	implement, space
spatula	spatula	kitchen_utensil
mug	mug	person
dish	dish	adult, tableware, food, antenna, female
plate	plate	cut, course, artifact, body_part, tableware, layer
bowl	bowl	artifact, solid
shaker	shaker	person, causal_agent
teaspoon	teaspoon	container, cutlery
saucepan	saucepan	kitchen_utensil
pan	pan	kitchen_utensil
glass	glass	drug, matter
tumbler	tumbler	athlete, container
coasters	coasters	traveler, inhabitant
muffin tin	tin	chemical_element
hand mixer	mixer	food,equipment,liquid
cake tester	cake	artifact,food,nutriment
cafetiere	block	group, region, copy, whole
saltball	pan	kitchen_utensil
pot	pot	fixture,resistor,soft_drug,kitchen_utensil
cooker	cooker	kitchen_utensil
chopper	chopper	aircraft, bone, edge_tool
blender	blender	kitchen_utensil
grinder	grinder	machine, snack_food, bone
steamer	steamer	kitchen_utensil
jar	jar	white_goods
fryer	fryer	poultry
scoop	scoop	radiation
mortar	mortar	artifact
nutribullet	container	artifact
sandwich press	sandwich	dish
corkscrew	corkscrew	opener

In the case where the parent of the object name cannot be found in WordNet, the keyphrase search is employed in the Web Documents search. The keyphrase is formed in the format as “object name keyword” as given in Table 6.17. The results depend on the richness of the information in the Web Documents. There are 35 object names in the experiment. The results show 37.14% of correct category name identification from “is” keyword. (The highlighted category names are the correct category names.)

Table 6.17 The keyphrase with the different keywords

	Keywords				
Object name	**is**	**is or contain**	**is or including**	**contain or including**	**all keywords**
mochi	rice	form	rice		form
daifuku	strawberry	link	strawberry		link
ice cream cake	cream	cake	ice		cake
flaugnarde	french	cherry	french	person	container
kronenbourg	brewery	page	people	son	margin
prosecco	wine	external	wine	grape	wine
sauvignon	wine	wine	wine	example	wine
jameson	whiskey	whiskey	whiskey		whiskey
mojito	juice	cocktail	cocktail		page
yakult	container	sugar	bottle	children	container
instant noodle	noodle	noodle	noodle	noodle	noodle
remoulade	shrimp	social	shrimp		social
ramen	bowl	width			container
miso	condiment	condiment	condiment	condiment	condiment
sauerkraut	nutriment	nutriment	nutriment	nutriment	nutriment
basmati rice	rice	rice	indian	indian	container
konjac	corm	fiber	gum	gum	fiber
tteokbokki	rice	container	rice	spain	container
chikuwa	skin	article	skin		article
okonomiyaki	hiroshima	hiroshima	hiroshima	highlight	hiroshima
nabemono	article	article	min	add-on	article
pfefferpotthast	span	span	story		span
spanferkel	pig				
fajita	container	container	stalk	stalk	container
cough syrup	cough	container	cough	cough	cough
inhalator	inhalator	container	container	container	container
nicorette	auto	character	arab	arab	character
senokot	senna	senna	taking	widget	senna
windsetlers	storage	container	arab	arab	storage
revitalens	lense	character	lense	lense	lense
germolene	state	state	isle	isle	state
nicolites cartomiser menthol	cigarette	cigarette			cigarette
linctus	medicine	codeine	medicine	medicine	medicine
nutribullet	container	container	work	turn	container
sandwich press	sandwich	block	sandwich	sandwich	block
Correctness	13	10	10	4	10
Percent	37.14%	28.57%	31.43%	14.29%	31.43%

6.2.3 Analysis of Results

The Automatic Ontology process changes object names and their property values to concept names, relations between concepts, instances, property names and values. In order to analyse the results, the object names are compared with the category name by using semantic similarity.

The semantic similarity measures the similarity score between two concepts. The two concepts are connected through "is-a" relations (Wong et al., 2012). The semantic similarity is calculated from Wu and Palmer's (1994) measurement as equation 6.1.

$$Sim(C_1, C_2) = \frac{2 \times depth(LCS(c_1 c_2))}{x_1 + x_2 + 2 \times depth(LCS(c_1, c_2))} \qquad (6.1)$$

Sim (c_1, c_2) is the semantic similarity score between concept 1 (c_1) and concept 2 (c_2). The Least Common Ancestor (LCS) is the concept that subsumes both terms. Depth (LCS (c_1, c_2)) is the depth from root concept to LCS of c_1 and c_2. Wu and Palmer count the number of "is-a" relations from c_1 to LCS and c_2 to LCS as x_1 and x_2, respectively.

The reasons for using the semantic similarity between two concepts are as follows:

- The measure is suitable to measure an "is-a" structure, as Robot Ontology is built from "is-a" relations from WordNet.
- The result of the semantic similarity score indicates the relatedness of the concepts. The greater semantic similarity score of two concepts indicates the greater links between the concepts and the more closely related they are. The semantic similarity score ranges between 0 and 1.

The framework assumes the concept name in Robot Ontology as concept name (c_1) in WordNet and the category name as concept name (c_2) in WordNet. It calculates the semantic similarity between c_1 and c_2 as equation 6.1. In the case where the semantic similarity between the two concept names is above the threshold of 0.5 in the experiment, the two concept names are related.

The semantic similarities between the concept names from the ten input files and their category names are given in Table 6.18. The semantic similarity scores are above the threshold for all concept names.

Table 6.18 Semantic similarity between concept names from the ten input files and their category names

Object name	Category name	Concept name	Semantic similarity
biscuit	pastry	biscuit	0.8
pepsi	drink	pepsi	0.8
beer	drink	beer	0.8
cocktail	drink	cocktail	0.8
milk	drink	milk	0.92
gravy	food	gravy	0.77
popcorn	food	popcorn	0.71
konjac	fruit and vegetable	corm	0.66
cheese	perishable	cheese	0.8
jam	perishable	jam	0.56

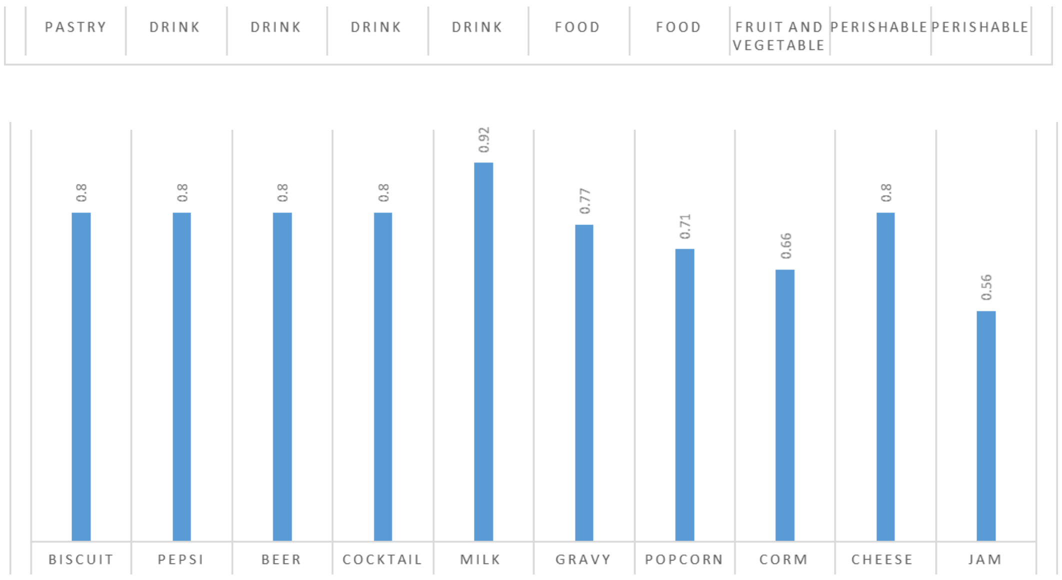


Figure 6.8 Semantic similarity between concept names from the ten input files and their category names

For example, the semantic similarity score between the "biscuit" concept name and the "pastry" category name is 0.8. This means that the "biscuit" concept name relates to the "pastry" category name, as the semantic similarity score is close to 1, as given in Figure 6.8. In the case where the semantic similarity score between the concept name and the category name is close to 1, the algorithm correctly creates the concept name in the category name.

The semantic similarity between the pastry category name and concept names are created from the proposed algorithm. The average of the semantic similarity score is 0.73 and the standard deviation is 0.08. The algorithm searches for the concept names for "mochi", "daifuku", "ice cream cake" and "flaugnarde" object names. The results of concept name from the algorithm are "rice", "strawberry", "cream" and "strawberry", and the semantic similarity scores are 0.71, 0.62, 0.71 and 0.62, respectively. The definition of "mochi" and "daifuku" are a kind of rice cake, the definition of "ice cream cake" is a kind of cake and the definition of "flaugnarde" is a baked French dessert.

However, the algorithm selects the "strawberry" category name for the "daifuku" and "flaugnarde" object names. The semantic similarity between "strawberry" and "daifuku", and "strawberry" and "flaugnarde" are 0.62.

Figure 6.9 shows the semantic similarity between the pastry category name and the concept names. Overall, the semantic similarity scores are above the threshold. The low standard deviation means that concept the names are related to the pastry category name. In the case of high semantic similarity scores of all concept names and the low standard deviation score, all concept names are arranged into the related category name. This means that all concept names are arranged correctly into the pastry category name.

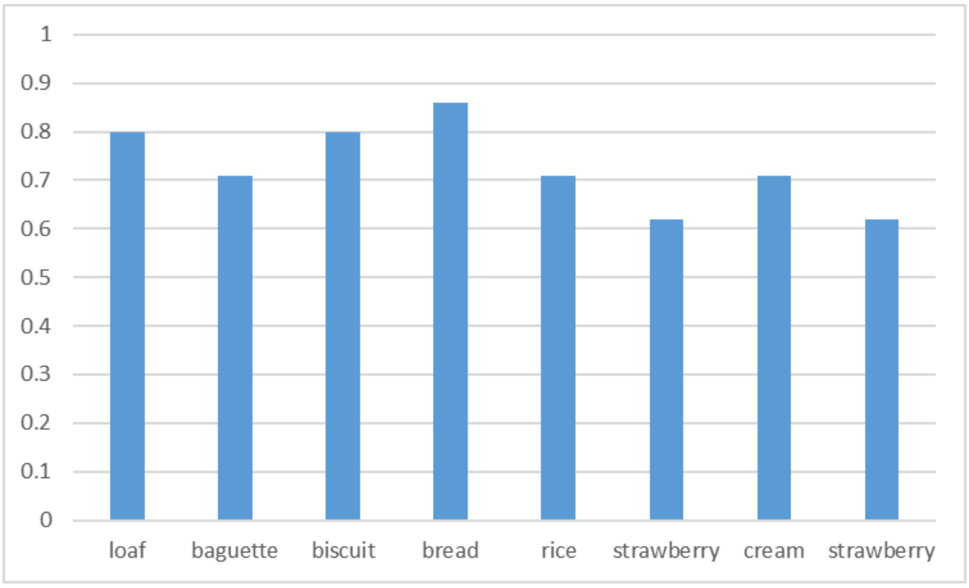


Figure 6.9 Semantic similarity between concept names and pastry

The semantic similarity between the drink category name and the concept names are created from the proposed algorithm. The average of the semantic similarity score is 0.78 and the standard deviation is 0.16.

Figure 6.10 shows the semantic similarity between the drink category name and the concept names. The semantic similarity between the drink category name and the concept names varies. There are three low values at brewery (0.29), squash (0.43) and container (0.33).

The algorithm searches for the concept name of “kronenbourg”, “squash”, and “yakult” object names. The results of the concept name from the algorithm are “brewery”, “squash”, and “container”, and the semantic similarity score are 0.29, 0.43 and 0.33, respectively. This means that there are outlier concept names in the drink category name. “kronenbourg” is a brand of beer, “squash” is a concentrated liquid that is made from fruit and “yakult” is a brand of drink.

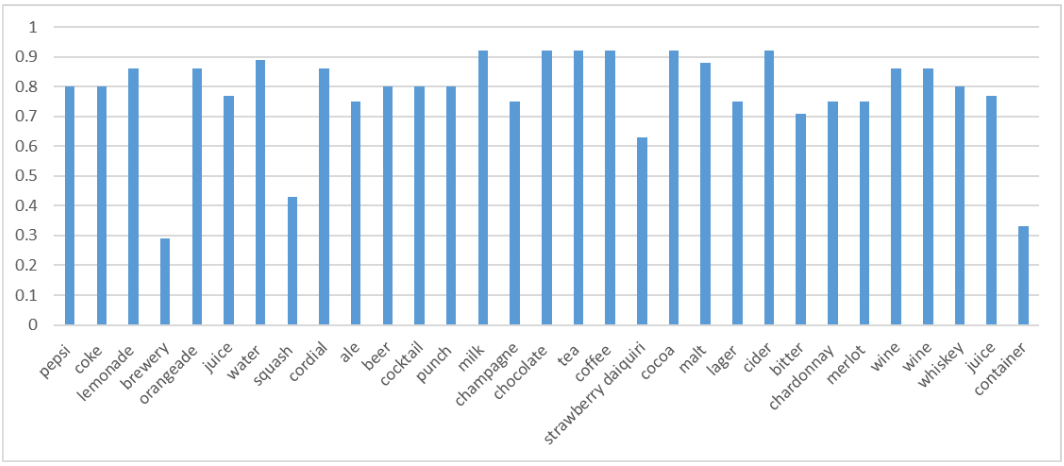


Figure 6.10 Semantic similarity between concept names and drink

The semantic similarity between the food category name and the concept names are created by the proposed algorithm. The average of the semantic similarity score is 0.73 and the standard deviation is 0.15.

Figure 6.11 shows the semantic similarity between the food category name and the concept names. The semantic similarity between the food category name and the concept names varies. There are two low values at bowl (0.46) and container (0.33). This means that there are outlier concept names in the food category name.

The "ramen" is a Japanese noodle soup dish and "basmati rice" is a kind of rice. The algorithm selects the "bowl" concept name for the "ramen" object name and selects the "container" concept name for the "basmati" object name from Web Documents.

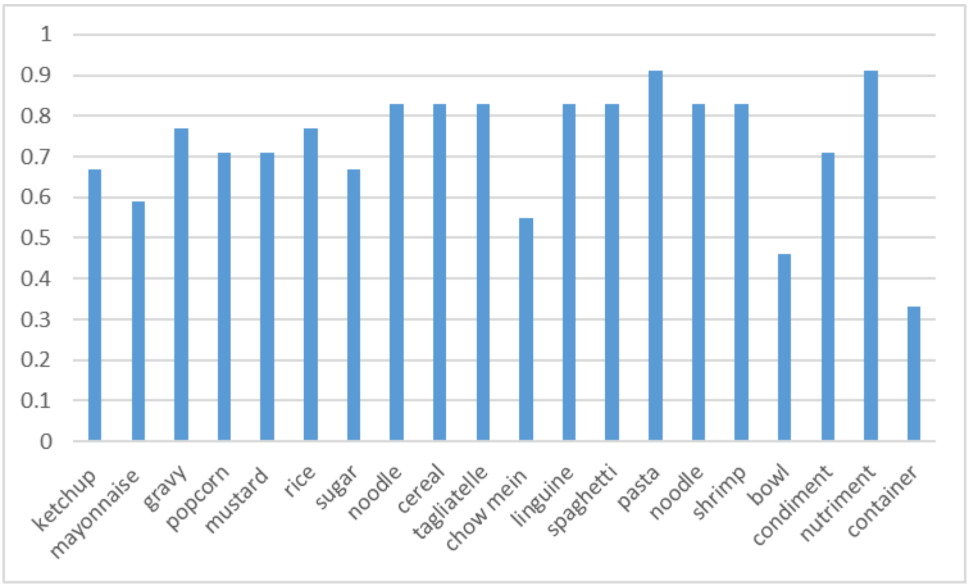


Figure 6.11 Semantic similarity between concept names and food

The semantic similarity between the fruit and vegetable category name and the concept names are created by the proposed algorithm. The average of the semantic similarity score is 0.80 and the standard deviation is 0.07. Figure 6.12 shows the semantic similarity between the fruit and vegetable category name and the concept names. Overall, the semantic similarity scores are nearly the same high score for all concept names. The low standard deviation means that, the concept names are related to the fruit and vegetable category name. In the case of high semantic similarity scores of all concept names and the low standard deviation score, all the concept names are arranged into the related category name. This means that all the concept names are correctly arranged into the fruit and vegetable category name.

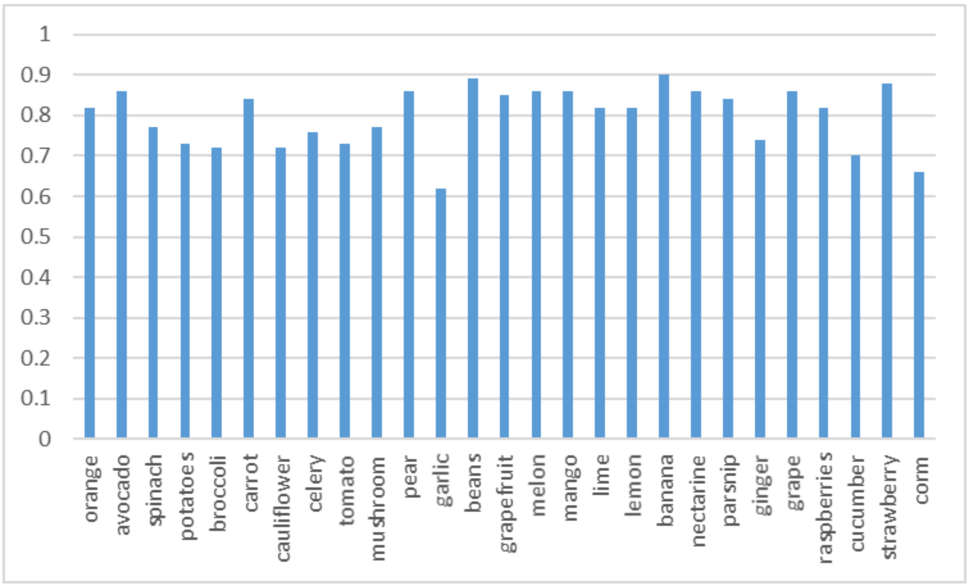


Figure 6.12 Semantic similarity between concept names and fruit and vegetable

The semantic similarity between the meat and fish category name and the concept names are created by the proposed algorithm. The average of the semantic similarity score is 0.75 and the standard deviation is 0.11. The low standard deviation means that the concept names are related to the meat and fish category name.

Figure 6.13 shows the semantic similarity between the meat and fish category name and the concept names. The semantic similarity between the meat and fish category name and concept names varies. There is one low value at mince (0.46). This means that there are outlier concept names in the meat and fish category name.

The algorithm selects the “mince” concept name for the “mince” object name but the semantic similarity score between “mince” and “meat”, and “mince” and “fish” is 0.46.

The semantic similarity between the perishable category name and the concept names are created by the proposed algorithm. The average of the semantic similarity score is 0.58 and the standard deviation is 0.21.

Figure 6.14 shows the semantic similarity between the perishable category name and the concept names. The semantic similarity between the perishable category name and the concept names varies from the average of the semantic similarity score. There are eight low values at quiche (0.3), clementine (0.35), skin (0.43), hiroshima (0.3), article (0.31), span (0.29), pig (0.27) and container (0.29). This means that there are outlier concept names in the perishable category name.

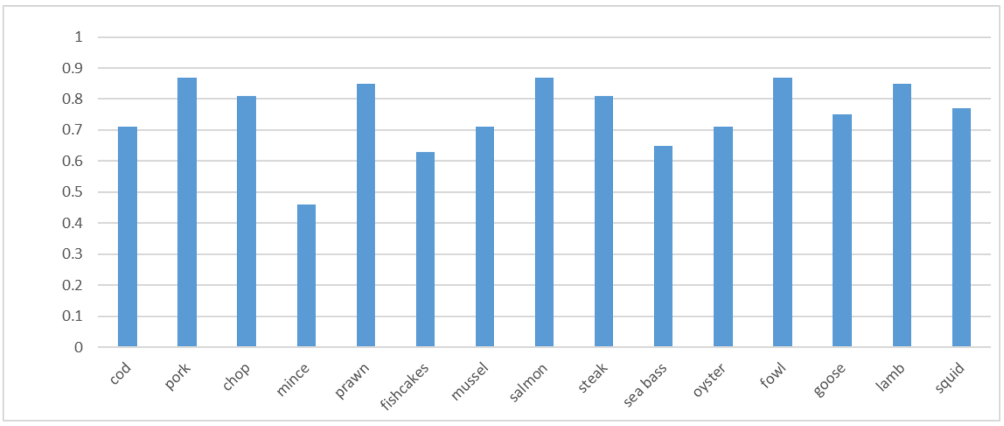


Figure 6.13 Semantic similarity between concept names and meat and fish

The “chikuwa” is a kind of food product. The algorithm selects the “skin” concept name for the “chikuwa” object name. “okonomiyaki” is a Japanese savoury pancake. The algorithm selects the “hiroshima” concept name for the “okonomiyaki” object name. “nabemono” is a hot pot dish. The algorithm selects the “article” concept name for the “nabemono” object name. “pfefferpotthast” is a kind of perishable. The algorithm selects the “span” concept name for the “pfefferpotthast” object name. “spanferkel” is roasted or grilled suckling pig. The algorithm selects the “pig” concept name for the “spanferkel” object name. “fajita” is a kind of perishable. The algorithm selects the “container” concept for the “fajita” object name.

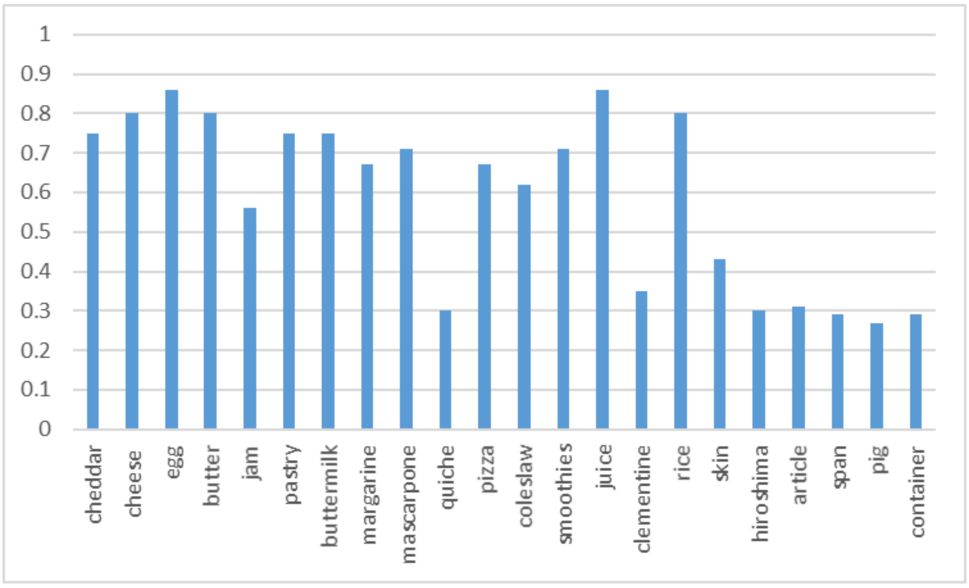


Figure 6.14 Semantic similarity between concept names and perishable

The semantic similarity between the product category name and the concept names are created by the proposed algorithm. The average of the semantic similarity score is 0.66 and the standard deviation is 0.07. The low standard deviation means that the concept names are related to the product category name.

Figure 6.15 shows the semantic similarity between the product category name and the concept names. Overall, the semantic similarity scores are nearly the same, high score for all the concept names. The low standard deviation means that the concept names are related to the product category name. In the case of high semantic similarity scores of all concept names and the low standard deviation score, all the concept names are arranged into the related category name. This means that all the concept names are correctly arranged into the product category name.

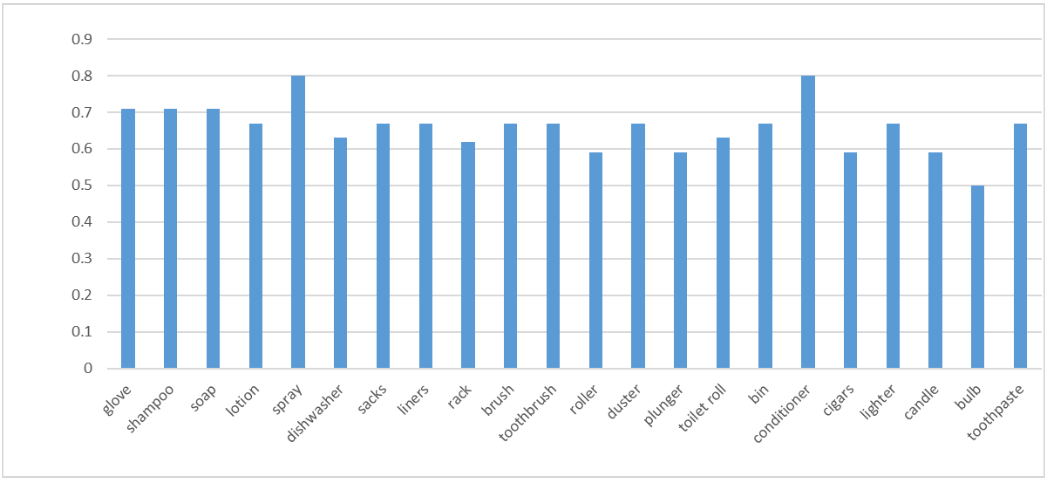


Figure 6.15 Semantic similarity between concept names and product

The semantic similarity between the medicine category name and the concept names are created by the proposed algorithm. The average of the semantic similarity score is 0.53 and the standard deviation is 0.25.

Figure 6.16 shows the semantic similarity between the medicine category name and the concept names. The semantic similarity between the medicine category name and the concept names varies from the average of the semantic similarity score. There are ten low values at cough (0.38), pastilles (0.44), syrup (0.47), plaster (0.46), bandage (0.4), inhalator (0.27), auto (0.24), senna (0.25), lense (0.27) and state (0.33). This means that there are outlier concept names in the medicine category name.

The “cough syrup” is a kind of medicine. The algorithm selects the “cough” concept name for the “cough syrup” object name. A “pastille” is a small sweet or lozenge. The algorithm selects the “skin” concept name for the “pastille” object name but the semantic similarity is 0.44. “syrup” is a thick sweet sticky liquid. The algorithm selects the “syrup” concept name for the “syrup” object name.

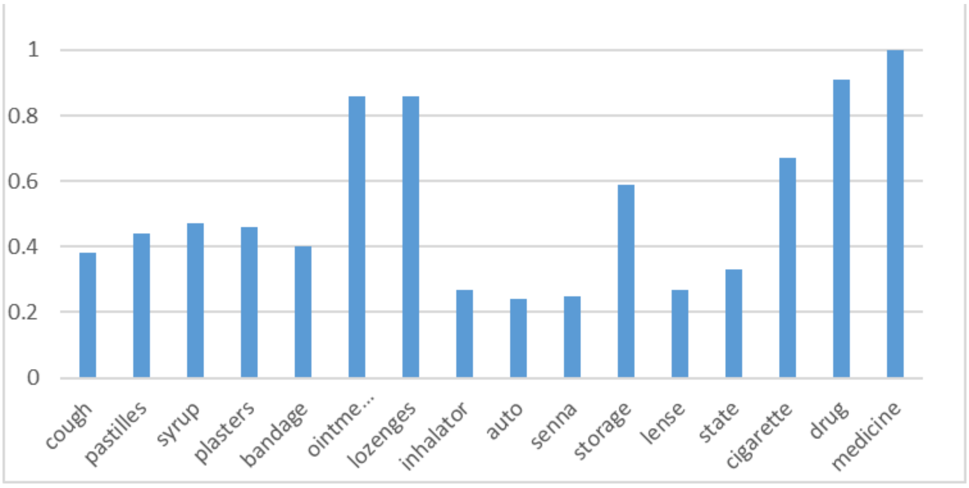


Figure 6.16 Semantic similarity between concept names and medicine

A “plaster” is a kind of a medical dressing. The algorithm selects the “plaster” concept name for the “plaster” object name but the semantic similarity is 0.46. A “bandage” is a piece of soft material that covers and protects an injured part of the body. The algorithm selects the “bandage” concept name for the “bandage” object name but the semantic similarity is 0.4. An “inhalator” is a breathing device. The algorithm selects the “inhalator” concept name for the “inhalator” object name but the semantic similarity is 0.27. “nicorette” is a brand of smoking control product. The algorithm selects the “auto” concept name for the “nicorette” object name.

The “senokot” is a brand of medicine. The algorithm selects the “senna” concept name for the “senokot” object name. Senna is a yellow flower used in a medicinal manner. “revitalens” is a brand of disinfecting solution for soft contact lenses. The algorithm selects the “lens” concept name for the “revitalens” object name. “germolene” is a brand of antiseptic product. The algorithm selects the “state” concept name for the “germolene” object name.

The semantic similarity between the kitchenware category name and the concept names are created by the proposed algorithm. The average of the semantic similarity score is 0.65 and the standard deviation is 0.17.

Figure 6.17 shows the semantic similarity between the kitchenware category name and concept names. The semantic similarity between the kitchenware category name and the concept names varies from the average of the semantic similarity score. There are three low values at baking tray (0.45), fryer (0.2) and sandwich press (0.21). This means that there are outlier concept names in the kitchenware category name.

A "baking tray" is a flat pan. The algorithm selects the "baking tray" concept name for the "baking tray" object name but the semantic similarity is 0.45. A "fryer" is a kind of kitchen appliance. The algorithm selects the "fryer" concept name for the "fryer" object name but the semantic similarity is 0.2. A "sandwich press" is a kind of kitchen appliance. The algorithm selects the "sandwich" concept name for the "sandwich press" object name.

Table 6.19 The overall score of nine categories

Category name	**Average**	**Standard deviation**	**Correct**	**Total**	**Percent**
pastry	0.73	0.08	8	8	100%
drink	0.78	0.16	28	31	90%
food	0.75	0.11	18	20	90%
fruit and vegetable	0.80	0.07	27	27	100%
meat and fish	0.74	0.12	14	15	93%
perishable	0.58	0.21	14	22	64%
product	0.65	0.08	22	22	100%
medicine	0.53	0.25	6	16	38%
kitchenware	0.68	0.15	36	39	92%
Summary			173	200	86.5%

The greater the semantic similarity score of two words, the more similar the meaning between two words. The overall score of correctness of the ontology, as given in Table 6.19, is designed for a human who applies the DIRAOF framework. It shows the validation of the concept hierarchy of the dataset. The overall score of correctness of the ontology shows the correctness at 86.5% for 200 random object names. The 13.5% is incorrect due to the description of a particular object name being missing from the Web Documents source. The disadvantage of Web Documents is that they have to be updated by humans to update the description of objects.

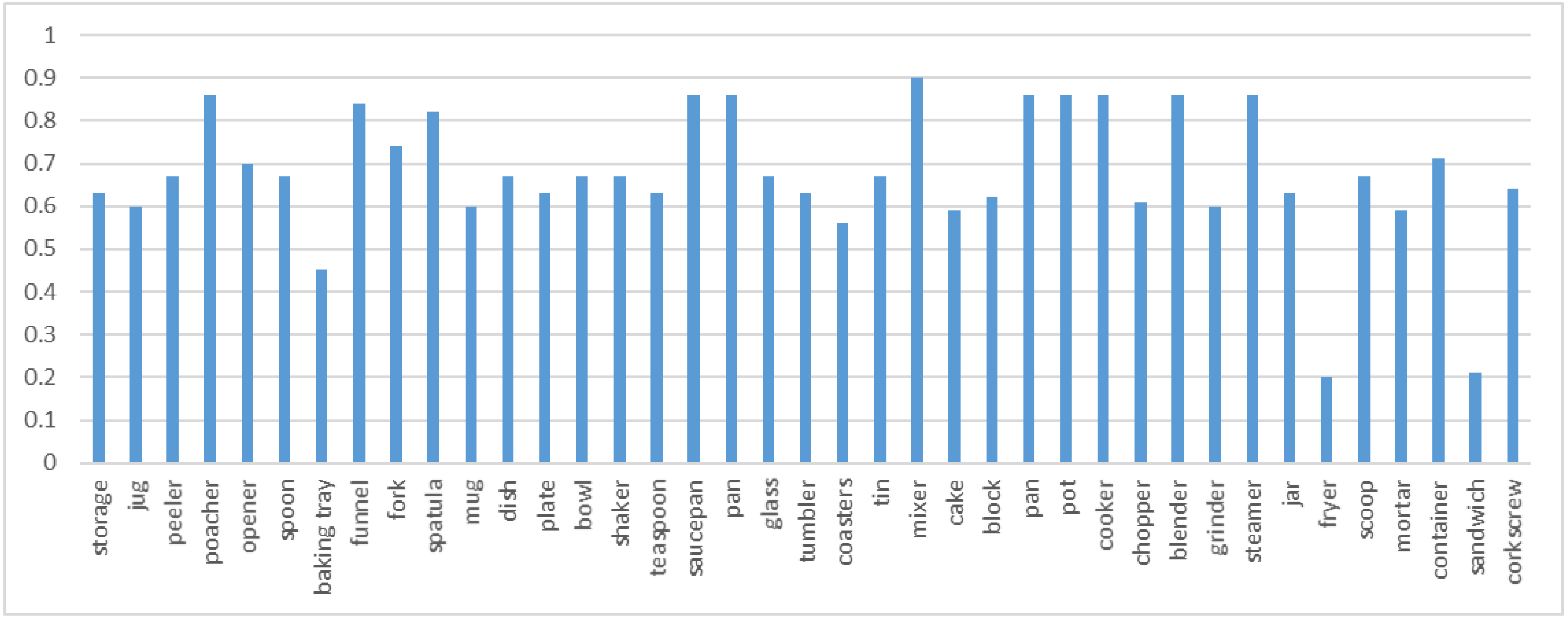


Figure 6.17 Semantic similarity between concept names and kitchenware category name

CHAPTER 7 CONCLUSION AND FURTHER WORK

This research develops techniques to enable domestic robots to automatically build up ontology components and to associate semantic knowledge with instances in an ontology.

7.1 Conclusion

This research investigated the effectiveness of using different keywords in information collection. It has been found that the keyword “is” is the most suitable one in finding a category name when searching for category names from Web Documents. The comparison among the different keywords is given in Table 6.17.

This research found the necessity for having “HasA” and “MadeOf” relations in a domestic robot ontology and developed the technique of creating the relations between concepts as the part of automatic ontology development. The technique searches for the relevant concepts in ConceptNet that may have “HasA” and “MadeOf” relations with the newly created concepts. The base URL is “http://conceptnet5.media.mit.edu/data/5.4/search”. The “HasA&start+ concept” and “MadeOf&start+ concept” arguments are set to specify the “HasA” and “MadeOf” relations in the way that “the newly created concept has a relevant concept” as a start parameter for “HasA” relation and “the newly created concept is made of a relevant concept” as a start parameter for “MadeOf”. In the case where the relevant concept can be found in the existing robot ontology, the relations of “HasA” and “MadeOf” are then created from the newly created concept and the existing concepts in the existing robot ontology.

This research also included semantic knowledge into the robot ontology. The semantic knowledge is represented with three properties, namely, “hasTime”, “hasLocation” and “hasProperty”. These property names are associated with instances of each concept in such a way that, given an object name, property values are picked up from the observed text files such the labels of the object, and the object, on the other hand, is viewed as an instance of the concept that the object belongs to. This research also developed a recency and frequency based and the NFile algorithm to calculate the location of objects based on the “hasLocation” and “hasTime” property names.

This research confirmed the feasibility of automatic robot ontology with the development of the integrated automatic robot ontology framework DIRAOF. Observing an object and its property values, the framework is able to search for information and to create a new concept, relations between the newly created concept with the existing concepts in the robot ontology, instances of the concept, property names and property values. It is also able to create semantic knowledge of the instances of concept.

7.2 Contributions

The main contributions of the research are the following.

Integrated automatic ontology for service robots

The automatic robot ontology is important for service robots that are deployed. It assigns the robots with the ability to continuously develop their ontology with limited human interventions. The framework can be applied to other robotic systems that work in unstructured and dynamic environments (section 4.2).

Discovery of the most effective keyword “is” for information collection

The research investigated the effectiveness of using different keywords in information collection. The “is” keyword gives the correct category name for an object name better than other keywords, as shown in the experiment. The “is” keyword supports robots to identify an object name that cannot be found in WordNet and to create the concept and an instance of the concept. Robots use the “is” keyword to search for the category name by selecting the noun with the highest frequency as the category name. The category name represents the general idea of that object name and it is created as a concept and an instance of the concept in the ontology.

“HasA” and “MadeOf” relations generation and inclusion

The “HasA” and “MadeOf” relations will support robots to handle vagueness contained in a human user’s commands (section 4.2.3). Human users give the commands such as to find what they need. For example, they want to have water to drink. Robots will need to link the command that has the word water with objects/instances on the ontology to support them to complete the task. The two relations “HasA” and “MadeOf” can tell the robots the objects/instances that contain water for drinking. The “HasA” and “MadeOf” relations between concepts can be applied for the query to search similar objects (section 5.3). For example, human users need to prepare dinner but they have limited condiments and foods. Robots can provide the relevant foods and condiments (instances of the “food” and “condiment” concepts) in the dynamic environment from their “HasA” and “MadeOf” relations between concepts.

Semantic knowledge creation

The inclusion of semantic knowledge into robots’ ontologies allows robots to handle uncertainly caused by environment sharing between human users and robots. The robots can use the historical data stored in NFile about the location of objects in terms of the “hasLocation” and “hasTime” property names to calculate recency and frequency in order to determine the location of the object the human user asks for (section 5.2).

7.3 Further Work

This study has focused on the development of the DIRAOF and on the implementation of the basic principle of the framework components. Further work will be needed on further development of the components.

Data Input process

The framework requires object names and property values of physical objects as text. However, homecare robots retrieve multiple sources of information from object perception. Problems will occur when the framework is utilised with a system that does not have text format as input. Therefore, the input process needs to be improved in further research. Image understanding, computer vision and object recognition will need to be added in further research. The Data Input process selects nouns and adjectives from labels of physical objects. However, nouns and adjectives are limited when creating relations between concepts. Therefore, the Data Input process needs to be improved in further research. An automatic method to identify nouns, verbs and adjectives needs to be developed.

Automatic Ontology process

Ontology development in Automatic Ontology develops the hierarchical structure of concepts. The ontology creation is an iterative process and often consumes time. Therefore, further research will be needed to find methods that reduce time consumption in the creation process. This might be to apply deep learning for ontology development process in further research, as deep learning being hierarchical learning for machine learning. It is based on learning representations of data with multiple processing layers. It will reduce the time for creating concepts and relations between concepts at large-scale in the domain of interest in further research.

Property Creation provides five predefined properties for sending to Semantic Knowledge Acquisition to associate information about instances. However, homecare robots require more information about instances in the environment. Therefore, other properties need to be added in further research. To achieve the aim, the object property learning will need to automatically identify and create object properties of objects in further research.

Data and Information Retrieval process

Robots may categorise objects in the wrong group because the category name is selected for an object name which relies on the rich information on the web and the keyword that is used to search for the category name. Therefore, the algorithm for searching unknown objects that cannot be found in WordNet needs to be improved in further work. To achieve the aim, the interaction with human users and the automatic translation into a machine-readable format would be a way to retrieve the category name of objects from the expert human in further research.

Semantic Knowledge Acquisition process

The Semantic Knowledge Acquisition process provides a method for locating the instances of a concept in a dynamic environment. However, it cannot guarantee that the location is the exact location of an object in a dynamic environment. Tracking Instances needs to be improved in further work. Logical reasoning would be an approach to integrate into Tracking Instances in further research.

Robot Ontology process

Robot ontology does not support service robots to exchange ontology and semantic knowledge with other systems. It cannot use the experience of other robots to create knowledge. Therefore, Robot Ontology needs to be improved in further work. Sharing of information and knowledge will be needed in further research.

Query and Result Evaluation processes

The Query process does not provide a user interface for human users to query Robot Ontology. It is difficult for users to enquire knowledge directly from Robot Ontology. Therefore, the Query process needs to be improved in further work in order to provide access to Robot Ontology to users. The user interface will be added in further research. Result Evaluation also does not provide access to the correctness of Robot Ontology to user. It is difficult for users to decide how good the dynamic ontology created in Robot Ontology. Therefore, Result Evaluation needs to be improved by adding a user interface in further research.

References

Al-Arfaj, A. and Al-Salman, A. (2015) ‘Ontology Construction from Text: Challenges and Trends’, *International Journal of Artificial Intelligence and Expert Systems (IJAE)*, 6(2), pp. 15-26.

Alfaries, A. (2010) ‘Ontology learning for Semantic Web services’, Doctoral dissertation, Brunel University, UK.

Aubin, S. and Hamon, T. (2006) ‘Improving term extraction with terminological resources’, In *Proceeding of the Fifth International Conference on Advances in Natural Language Processing*, pp. 380-387.

Auger, A. and Barrière, C. (2008) ‘Pattern-based approaches to semantic relation extraction: A state-of-the-art’, *Terminology,* 14(1), pp. 1-19.

Barforush, A.A. and Rahnama, A. (2012) ‘Ontology learning: revisited’, *Journal of Web Engineering*, 11(4), pp. 269-289.

Berland, M. and Charniak, E. (1999) ‘Finding parts in very large corpora’, In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pp. 57-64.

Bisson, G., Nédellec, C. and Canamero, D. (2000) ‘Designing Clustering Methods for Ontology Building-The Mo'K Workbench’, In *Proceeding of the 14th European Conference on Ontology Learning,* pp. 13-19.

Bouguerra, A., Karlsson, L. and Saffiotti, A. (2008) ‘Monitoring the execution of robot plans using semantic knowledge’, *Robotics and Autonomous Systems*, 56(11), pp. 942-954.

Brants, T. and Franz, A. (2006) *Web 1T 5-gram Version 1*. Linguistic Data Consortium, Philadelphia.

Brewster, C., Iria, J., Zhang, Z., Ciravegna, F., Guthrie, L. and Wilks, Y. (2007) ‘Dynamic iterative ontology learning’, In *Proceedings of sixth International Conference on Recent Advances in Natural Language Processing*, pp. 93-97.

Buitelaar, P. and Cimiano, P. (2008) *Ontology learning and population: bridging the gap between text and knowledge*. IOS Press, Amsterdam.

Buitelaar, P., Cimiano, P. and Magnini, B. (2005) *Ontology learning from text: methods, evaluation and applications*. IOS Press, Amsterdam.

Care-O-bot 4. (2016) *Care-O-bot 4*. Available at: http://www.care-o-bot-4.de/. Last updated: 7 January 2016. (Accessed: 1st November 2016).

Carpenter, B. (1992) *The logic of typed feature structures: with applications to unification grammars, logic programs and constraint resolution*. Cambridge University Press.

Caraballo, S.A. and Charniak, E. (1999) ‘Determining the specificity of nouns from text’, In *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pp. 63-70.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. and Kuksa, P. (2011) ‘Natural language processing (almost) from scratch’, *The Journal of Machine Learning Research*, 12(1), pp. 2493-2537.

Chuang, S.L. and Chien, L.F. (2005) ‘Taxonomy generation for text segments: A practical web-based approach’, *ACM Transactions on Information Systems*, 23(4), pp. 363-396.

ConceptNet5**.** (2015) *ConceptNet 5 API.* Available at: https://github.com/commonsense/conceptnet5/wiki/API. Last updated: 7 December 2015. (Accessed: 1st February 2016).

D’Este, C. and Sammut, C. (2008) ‘Learning and generalising semantic knowledge from object scenes’, *Robotics and Autonomous Systems*, 56(11), pp. 891-900.

Dan, Z., Li, Z. and Hao, J. (2010) ‘Research on Semi-Automatic Domain Ontology Construction’, In *Proceedings of the Seventh IEEE International Conference on Web Information Systems and Applications (WISA)*, pp. 115-118.

Davis, R., Shrobe, H. and Szolovits, P. (1993) ‘What is a knowledge representation? *AI Magazine*, 14(1), pp. 17-33.

Dean, M., Schreiber, G., Bechhofer, S., Van Harmelen, F., Hendler, J. Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F. and Stein, L.A. (2004) ‘OWL web ontology language reference’, *W3C Recommendation*, Available at: http://www.w3.org/TR/2004/REC-owl-ref-20040210/.

Dellschaft, K. and Staab, S. (2006) ‘On how to perform a gold standard based evaluation of ontology learning’, In *Proceedings of the Fifth International Semantic Web Conference (ISWC)*, pp. 228-241.

Dellschaft, K. and Staab, S. (2008) ‘Strategies for the evaluation of ontology learning’, In *Proceedings of the 2008 Conference on Ontology Learning and Population: Bridging the Gap between Text and Knowledge, Frontiers in Artificial Intelligence and Applications*, pp. 253-272.

Embley, D.W., Jiang, Y. and Ng, Y.K. (1999) ‘Record-boundary discovery in Web documents’, In *ACM SIGMOD Record,* 28(2), pp. 467-478.

Evert, S. (2010) ‘Google Web 1T 5-Grams Made Easy (but not for the computer)’, In *Proceedings of the NAACL HLT 2010 Sixth Web as Corpus Workshop,* pp. 32-40.

Fahad, M. and Qadir, M.A. (2008) ‘A Framework for Ontology Evaluation’, *ICCS Supplement*, 354, pp. 149-158.

Gangemi, A. (2005) ‘Ontology design patterns for semantic web content’, In *Proceedings of the Fourth International Semantic Web Conference (ISWC)*, pp. 262-276.

Gangemi, A., Guarino, N. and Oltramari, A. (2001) ‘Conceptual analysis of lexical taxonomies: The case of WordNet top-level’, In *Proceedings of the ACM international conference on Formal Ontology in Information Systems*, pp. 285-296.

Gargouri, F. (2010) *Ontology Theory, Management and Design: Advanced Tools and Models: Advanced Tools and Models*. Hershey, New York.

Gómez-Pérez, A. and Corcho, O. (2002) ‘Ontology languages for the semantic web’, *IEEE Intelligent Systems*, 17(1), pp. 54-60.

Gómez-Pérez, A., Fernández-López, M. and Corcho, O. (2004) *Ontological Engineering.* Springer, New York.

Granger, E.H. (1984) ‘Aristotle on genus and differentia’, *Journal of the History of Philosophy*, 22(1), pp. 1-23.

Gruber, T.R. (1993) ‘A translation approach to portable ontology specifications’, *Knowledge Acquisition*, 5(2), pp. 199-220.

Gu, T., Wang, X.H. Pung, H.K. and Zhang, D.Q. (2004) ‘An ontology-based context model in intelligent environments’, In *Proceedings of conference on communication networks and distributed systems modelling and simulation*, pp. 270-275.

Haidegger, T., Barreto, M., Gonçalves, P., Habib, M.K., Ragavan, S.K.V., Li, H., Vaccarella, A., Perrone, R. and Prestes, E. (2013) ‘Applied ontologies and standards for service robots’, *Robotics and Autonomous Systems*, 61(11), pp. 1215-1223.

Han, J. (2009) ‘Building an efficient, scalable, and trainable probability-and-rule-based part-of-speech tagger of high accuracy’, Master’s dissertation, Athens, GA.

He, L. and Hou, H.Q. (2008) ‘Research on semi-automatic construction of domain ontology based on machine learning and clustering technique’, In *Proceedings of the Second IEEE international symposium on intelligent information technology application (IITA),* pp. 345-348.

Hearst, M.A. (1992) ‘Automatic acquisition of hyponyms from large text corpora’, In *Proceedings of the 14th conference on Computational linguistics,* pp. 539-545.

Herzog, A., Jacobi, D. and Buchmann, A. (2008) ‘A3ME-an Agent-Based middleware approach for mixed mode environments’, In *Proceeding of the Second IEEE International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM),* pp. 191-196.

Hertzberg, J. and Saffiotti, A. (2008) ‘Using semantic knowledge in robotics’, *Robotics and Autonomous Systems,* 56(11), pp. 875-877.

Holzapfel, H., Neubig, D. and Waibel, A. (2008) ‘A dialogue approach to learning object descriptions and semantic categories’, *Robotics and Autonomous Systems,* 56(11), pp. 1004-1013.

Hjelm, H. (2009) ‘Cross-language ontology learning: Incorporating and exploiting cross-language data in the ontology learning process’, Ph.D. dissertation, Stockholm University.

International Federation of Robotics. (n.d.) *Service Robots*. Available at: http://www.ifr.org/service-robots. (Accessed: 1st February 2016)

Jakus, G., Milutinovic, V., Omerovic, S. and Tomazic, S. (2013) *Concepts, Ontologies, and Knowledge Representation*. Springer, New York.

Jasper, R. and Uschold, M. (1999) ‘A framework for understanding and classifying ontology applications’, In *Proceedings 12th international Workshop on Knowledge Acquisition, Modelling, and Management KAW,* pp. 16-21.

Ji, Z., Qiu, R., Noyvirt, A., Soroka, A., Packianather, M., Setchi, R., Li, D. and Xu, S. (2012) ‘Towards automated task planning for service robots using semantic knowledge representation’, In *Proceedings of the Tenth IEEE International Conference on Industrial Informatics (INDIN),* pp. 1194-1201.

Keizer, N.D., Abu-Hanna, A. and Zwetsloot-Schonk, J.H.M. (2000) ‘Understanding terminological systems I: terminology and typology’, *Methods of Information in Medicine*, 39(1), pp. 16-21.

Khoo, C.S. and Na, J.C. (2006) ‘Semantic relations in information science’, *Annual Review of Information Science and Technology*, 40(1), pp. 157-228.

Lam, T.N., Lee, H., Mayama, K. and Mizukawa, M. (2012) ‘Evaluation of commonsense knowledge for intuitive robotic service’, In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA),* pp. 3679-3684.

Lemaignan, S. (2012) ‘Grounding the interaction: knowledge management for interactive robots’, Doctoral dissertation, CNRS-Laboratoire d’analyse et d’architecture des systèmes, Technische Universität München.

Lemaignan, S., Ros, R., Mösenlechner, L., Alami, R. and Beetz, M. (2010) ‘ORO, a knowledge management platform for cognitive architectures in robotics’, In *Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* pp. 3548-3553.

Lenat, D.B. (1995) ‘CYC: A large-scale investment in knowledge infrastructure’, *Communications of the ACM,* 38(11), pp. 33-38.

Lenat, D.B., Guha, R.V., Pittman, K. Pratt, D. and Shepherd, M. (1990) ‘Cyc: toward programs with common sense’, *Communications of the ACM*, 33(8), pp. 30-49.

Lim, G.H., Suh, I.H. and Suh, H. (2011) ‘Ontology-based unified robot knowledge for service robots in indoor environments’, *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 41(3), pp. 492-509.

Lin, D. and Pantel, P. (2001) ‘DIRT- discovery of inference rules from text’, In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 323-328.

López, M.F., Gómez-Pérez, A., Sierra, J.P. and Sierra, A.P. (1999) ‘Building a chemical ontology using methontology and the ontology design environment’, *IEEE Intelligent Systems*, 14(1), pp. 37-46.

Maedche, A. and Staab, S. (2000) ‘Discovering conceptual relations from text’ In *Proceeding of the 13th European Conference on Artificial Intelligence*, pp. 321-325.

Manning, C.D., Raghavan, P. and Schütze, H. (2008) *Introduction to information retrieval*. Cambridge University Press.

Martinez, A. and Fernández, E. (2013) *Learning ROS for robotics programming*. Packt Publishing, Birmingham, UK.

Martinet, P. and Patin, B. (2008) ‘Proteus: A platform to organise transfer inside French robotic community’, In *Proceeding of the Third National Conference on Control Architectures of Robots (CAR),* pp. 1-10.

Maynard, D., Funk, A. and Peters, W. (2009) ‘Using lexico-syntactic ontology design patterns for ontology creation and population’, In *Proceedings of the Workshop on Ontology Patterns,* pp. 39-52.

McEnery, T. and Wilson, A. (2001) *Corpus linguistics: An introduction.* Edinburgh University Press.

Mehlhorn, K. and Sanders, P. (2008). *Algorithms and data structures: The basic toolbox*. Springer Science & Business Media, Berlin.

Meijer, K., Frasincar, F. and Hogenboom, F. (2014) 'A semantic approach for extracting domain taxonomies from text', *Decision Support Systems*, 62, pp. 78-93.

Miller, G.A. (1995) 'WordNet: a lexical database for English', *Communications of the ACM*, 38(11), pp. 39-41.

Moldovan, D., Clark, C., Harabagiu, S. and Hodges, D. (2007) 'Cogex: A semantically and contextually enriched logic proverb for question answering', *Journal of Applied Logic*, 5(1), pp. 49-69.

Murdock, J., Buckner, C. and Allen, C. (2010) *Evaluating dynamic ontologies*, In *Knowledge Discovery, Knowledge Engineering and Knowledge Management*. Springer, Berlin, pp. 258-275.

Navigli, R. and Velardi, P. (2006) 'Ontology enrichment through automatic semantic annotation of on-line glossaries', In *Proceedings of the 15th International Conference on Managing Knowledge in a World of Networks,* pp. 126-140.

Navigli, R. and Velardi, P. (2008) 'From glossaries to ontologies: Extracting semantic structure from textual definitions', In *Proceedings of the 2008 Conference on Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, pp. 71-87.

Ngo, T.L., Lee, H. and Mizukawa, M. (2011) 'Automatic building robot technology ontology based on basic-level knowledge', *Journal of Robotics and Mechatronics*, 23(4), pp. 515-522.

Ngo, T.L., Ukai, K. and Mizukawa, M. (2010) ‘Development and evolution of RT ontology for automatic service generation system in Kukanchi’, In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3465-3470.

Novak, J.D. and Cañas, A.J. (2008) *The theory underlying concept maps and how to construct and use them*. Technical Report. Institute for Human and Machine Cognition, Pensacola, FL.

Noy, N.F. and McGuinness, D.L. (2001) ‘Ontology development 101: A guide to creating your first ontology’, Knowledge Systems Laboratory Tech report KSL-01-05, Stanford University, Palo Alto, CA.

Ogden, C.K. and Richards, I.A. (1923) *The Meaning of Meaning: A Study of the Influence of Language upon Thought and of the Science of Symbolism*. Routledge & Kegan Paul, London.

OWL 2. (2016) *OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition).* Available at: https://www.w3.org/TR/2012/REC-owl2-syntax-20121211. (Accessed: 1st February 2016)

Paull, L., Severac, G., Raffo, G.V., Angel, J.M., Boley, H., Durst, P.J., Gray, W., Habib, M., Nguyen, B.X., Ragavan, S.V. and Sajad Saeedi, G. (2012) ‘Towards an ontology for autonomous robots’, In *Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1359-1364.

PR2. (2016) *PR2*. Available at: http://www.willowgarage.com/pages/pr2/overview. Last updated: 19 August 2015. (Accessed: 1st November 2016).

Presutti, V. and Gangemi, A. (2008) ‘Content ontology design patterns as practical building blocks for web ontologies’, In *Proceedings of* 27*th international conference on conceptual modelling*, pp. 128-141.

Ribeiro de Azevedo, R., Freitas, F., Rocha, R.G., Alves de Menezes, J.A., de Oliveira Rodrigues C.M. and Silva, G.D.F. (2014) ‘An Approach for Learning and Construction of Expressive Ontology from Text in Natural Language’, In *Proceeding of the IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, pp. 149-156.

Rockel, S., Neumann, B., Zhang, J., Dubba, K.S.R., Cohn, A.G., Konečný, Š., Mansouri, M., Pecora, F., Saffiotti, A., Günther, M. and Stock, S. (2013) ‘An ontology-based multi-level robot architecture for learning from experiences’, In *Designing Intelligent robots: Reintegrating AI II.* AAAI Spring Symposium-Technical Report, pp. 52-57.

Sánchez, D. (2009) ‘Domain ontology learning from the web’, *The Knowledge Engineering Review*, 24(4), pp. 413-413.

Sanderson, M. and Croft, B. (1999) ‘Deriving concept hierarchies from text’, In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval,* pp. 206-213.

Schaaf, T. (2001) ‘Detection of OOV words using generalized word models and a semantic class language model’, In *Proceedings of Eurospeech*, pp. 2581-2584.

Schutz, A. and Buitelaar, P. (2005) ‘Relext: A tool for relation extraction from text in ontology extension’, In *Proceedings of the fourth International Semantic Web Conference (ISWC),* pp. 593-606.

Sellami, Z., Camps, V. and Aussenac-Gilles, N. (2013) ‘DYNAMO-MAS: a multi-agent system for ontology evolution from text’, *Journal on Data Semantics*, 2(2-3), pp. 145-161.

Smith, B. (2004) ‘Beyond concepts: ontology as reality representation’, In *Proceedings of the third international conference on formal ontology in information systems (FOIS),* pp. 73-84.

Socher, R., Huval, B., Manning, C.D. and Ng, A.Y. (2012) ‘Semantic compositionality through recursive matrix-vector spaces’, In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 1201-1211.

Sowa, J.F. (2000) ‘*Knowledge Representation: Logical, Philosophical, and Computational Foundations*’, Brooks Cole, Pacific Grove, CA.

Speer, R. and Havasi, C. (2012) Representing general relational knowledge in ConceptNet 5. In *Proceeding of the International conference on language resources and evaluation (LREC),* pp. 3679–86.

Suchanek, F.M., Ifrim, G. and Weikum, G. (2006) ‘Combining linguistic and statistical analysis to extract relations from web documents’, In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining,* pp. 712-717.

Suh, I.H., Lim, G.H., Hwang, W. Suh, H., Choi, J.H. and Park, Y.T. (2007) ‘Ontology-based multi-layered robot knowledge framework (OMRKF) for robot intelligence’, In *Proceeding of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 429-436.

Sun, Y., Bo, L. and Fox, D. (2014) ‘Learning to identify new objects’, In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA),* pp. 3165-3172.

Taylor, A., Marcus, M. and Santorini, B. (2003) ‘The Penn treebank: an overview’, *Text, Speech and Language Technology*, 20, pp. 5-22.

Tempich, C., Simperl, E. and Vrandecic, D. (2008) ‘A methodology for ontology learning’, In *Proceedings of the Conference on Ontology Learning and Population: Bridging the Gap between Text and Knowledge*, pp. 225-249.

Tenorth, M. (2011) ‘Knowledge processing for autonomous robots’, Doctoral dissertation, Technische Universität München.

Tenorth, M. and Beetz, M. (2009) ‘KnowRob—knowledge processing for autonomous personal robots’, In *Proceeding of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4261-4266.

Tenorth, M. and Beetz, M. (2013) ‘KnowRob: A knowledge processing infrastructure for cognition-enabled robots’, *The International Journal of Robotics Research*, 32(5), pp. 566-590.

Tenorth, M., Perzylo, A.C., Lafrenz, R. and Beetz, M. (2012) ‘The RoboEarth language: Representing and exchanging knowledge about actions, objects, and environments’, In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1284-1289.

Ukai, K., Ando, Y. and Mizukawa, M. (2009) ‘Investigation of user RT-service generation system design’, In *Proceedings of the IEEE International Conference on Fuzzy Systems*, pp. 1486-1491.

Ullman, J.D., and Widom, J. (2014) *A First Course in Database Systems: Pearson New International Edition*. (3rd edn.) Pearson Higher Education, London.

Uschold, M. ‘Building ontologies: Towards a unified methodology’, In *Proceeding of the 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems*, pp. 1-20.

Uschold, M. and King, M. ‘Towards a methodology for building ontologies’, In *Proceedings of International Joint Conference of Artificial Intelligence Workshop on Basic Ontological Issues in Knowledge Sharing*, pp. 15-30.

Velardi, P., Fasolo, M. and Pazienza, M.T. ‘How to encode semantic knowledge: a method for meaning representation and computer-aided acquisition’, *Computational Linguistics*, 17(2), pp. 153-170.

Velardi, P. and Pazienza, M.T. (1989) ‘Computer aided interpretation of lexical cooccurrences’, In *Proceedings of the 27th annual meeting on Association for Computational Linguistics*, pp. 185-192.

Völker, J., Hitzler, P. and Cimiano, P. (2007) ‘Acquisition of OWL DL axioms from lexical resources’, In *Proceedings of the fourth European Semantic Web Conference (ESWC),* pp. 670-685.

Völker, J. and Rudolph, S. (2008) ‘Lexico-logical acquisition of OWL DL axioms’, In *Proceedings of the sixth International Conference on Formal Concept Analysis (ICFCA)*, pp. 62-77.

Waibel, M., Beetz, M., Civera, J., d’Andrea, R., Elfring, J., Galvez-Lopez, D., Haussermann, K., Janssen, R., Montiel, J.M.M., Perzylo, A. and Schießle, B. (2011) ‘A world wide web for robots’, *IEEE Robotics & Automation Magazine*, 18(2), pp. 69-82.

Wang, X.H., Zhang, D.Q., Gu, T. and Pung, HK. (2004) ‘Ontology based context modelling and reasoning using OWL’ In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, pp. 18-22.

Wong, W., Liu, W. and Bennamoun, M. (2012) ‘Ontology learning from text: A look back and into the future’, *ACM Computing Surveys (CSUR)*, 44(4), pp. 20-36.

Yu, X. and Shen, G. (2013) ‘Research on Semi-Automatic Domain Ontology Construction Framework Based on Web Crawler’, *In Proceeding of the International Conference on Computer, Networks and Communication Engineering (ICCNCE)*, pp. 413-416.

Zhou, W., Liu, Z., Zhao, Y., Xu, L., Chen, G., Wu, Q., Huang, M.L. and Qiang, Y. (2006) ‘A semi-automatic ontology learning based on WordNet and event-based natural language processing’, In *Proceedings of IEEE International Conference on Information and Automation (ICIA)*, pp. 240-244.

Appendix A: "is-a", "HasA" and "MadeOf" relations between concepts and parents of concept

The pastry category name

object name	Concepts and Relations			
	is-a		HasA	MadeOf
loaf	loaf	solid		
baguette	baguette	white_bread		
	white_bread	starches		
	starches	food		
biscuit	biscuit	baked_goods		
	baked_goods	solid		
	biscuit	bread		
bread	bread	food		grain
mochi	rice	grass		
	grass	physical_entity		
	rice	foodstuff		
	foodstuff	substance		
	foodstuff	physical_entity		
	rice	writer		
	writer	person		
	person	physical_entity		
daifuku	strawberry	edible_fruit		
	edible_fruit	reproductive_structure		
	reproductive_structure	plant_part		
	plant_part	whole		
	edible_fruit	food		
	strawberry	vascular_plant		
	vascular_plant	organism		
	organism	whole		
ice cream cake	cream	foodstuff		
flaugnarde	strawberry	edible_fruit		

The drink category name

object name	Concepts and Relations			
	is-a		HasA	MadeOf
pepsi	pepsi	soft drink		
	soft drink	liquid		
	liquid	matter		
	soft drink	food		
coke	coke	substance		
lemonade	lemonade	beverage		
	beverage	substance		
brewery	brewery	artifact		
orangeade	orangeade	beverage		
juice	juice	food		
	juice	electrical_phenomenon		
	electrical_phenomenon	natural_phenomenon		
	natural_phenomenon	physical_entity		
	juice	body_substance		
	body_substance	matter		
water	water	physical_entity		
squash	squash	athletic_game		
	athletic_game	activity		
	activity	physical_entity		
	squash	produce		
	produce	solid		
cordial	cordial	drug_of_abuse		
	drug_of_abuse	agent		
	agent	physical_entity		
	cordial	beverage		
ale	ale	brew		
	brew	drug_of_abuse		
	brew	beverage		
beer	beer	alcohol	alcohol	
	beer	alcohol	water	
cocktail	cocktail	course		
	course	food		
	food	matter		
	matter	physical_entity		
	cocktail	alcohol		
	alcohol	food		
	alcohol	drug		
	alcohol	fluid		
	drug	physical_entity		
	fluid	physical_entity		

object name	Concepts and Relations			
	is-a		HasA	MadeOf
punch	punch	stroke		
	stroke	motion		
	motion	action		
	action	physical_entity		
	punch	alcohol		
	punch	implement		
	implement	artifact		
milk	milk	food	water	
	milk	stream		
	stream	thing		
	stream	physical_entity		
	thing	physical_entity		
	milk	liquid		
	milk	body_substance		
	body_substance	physical_entity		
champagne	champagne	wine	alcohol	
	wine	beverage		
	champagne	geographical_area		
	geographical_area	location		
	location	physical_entity		
chocolate	chocolate	food		
	chocolate	solid		
	chocolate	liquid		
tea	tea	nutriment		
	nutriment	substance		
	tea	food		
	tea	liquid		
	tea	party		
	party	physical_entity		
	tea	flavorer		
	flavorer	foodstuff		
coffee	coffee	food		water
	coffee	liquid		
strawberry daiquiri	strawberry daiquiri	cocktail		
cocoa	cocoa	liquid		
	cocoa	food		
malt	malt	foodstuff		beer
	malt	drink		
	drink	physical_entity		
	malt	beer		

object name	Concepts and Relations			
	is-a		HasA	MadeOf
lager	lager	military_quarters		
	military_quarters	housing		
	housing	artifact		
	lager	brew		
cider	cider	liquid		
	cider	food		
bitter	bitter	property		
	property	physical_entity		
	bitter	beer		
chardonnay	chardonnay	wine		
	wine	drug_of_abuse		
	wine	beverage		
merlot	merlot	wine		
prosecco	wine	drug_of_abuse		
	wine	beverage		
sauvignon	wine	drug_of_abuse		
	wine	beverage		
jameson	whiskey	alcohol		
mojito	juice	food		
	juice	electrical_phenomenon		
	juice	body_substance		
yakult	container	artifact		

The food category name

object name	Concepts and Relations			
	is-a		HasA	MadeOf
ketchup	ketchup	ingredient		
mayonnaise	mayonnaise	condiment		
gravy	gravy	condiment		
	condiment	ingredient		
	ingredient	food		
	gravy	event		
	event	physical_entity		
	gravy	foodstuff		
popcorn	popcorn	grain		
	grain	grass		
	grass	herb		
	herb	substance		
	herb	physical_entity		
	grass	physical_entity		
	grain	food		
	popcorn	cereal		
	cereal	food		
	substance	physical_entity		
mustard	mustard	vegetable		
	vegetable	food		
	mustard	herb		
	herb	living_thing		
rice	rice	grass		alcohol
	rice	foodstuff		
	rice	writer		
sugar	sugar	organic_compound		bread
	organic_compound	chemical		
	chemical	physical_entity		
	sugar	flavorer		
	sugar	molecule		
	molecule	thing		
	thing	physical_entity		
	sugar			pepsi
	sugar			chocolate

object name	Concepts and Relations			
	is-a		HasA	MadeOf
noodle	noodle	food		
	noodle	head		
	head	physical_entity		
cereal	cereal	food		
tagliatelle	tagliatelle	food		
chow mein	chow mein	nutriment		
linguine	linguine	food		
spaghetti	spaghetti	dish		
	dish	food		
	spaghetti	food		
pasta	pasta	nutriment		
	pasta	solid		
instant noodle	noodle	food		
remoulade	shrimp	person		
	shrimp	food		
ramen	bowl	artifact		
	bowl	solid		
miso	condiment	ingredient		
sauerkraut	nutriment	substance		
basmati rice	container	artifact		

The fruit and vegetable category name

object name	Concepts and Relations			
	is-a		HasA	MadeOf
orange	orange	stream		
	stream	thing		
	orange	color		
	color	property		
	property	physical_entity		
	color	substance		
	orange	coloring_material		
	coloring_material	substance		
avocado	avocado	produce		
spinach	spinach	vegetable		
	vegetable	food		
	spinach	herb		
potatoes	potatoes	vegetable		
	potatoes	foodstuff		
broccoli	broccoli	herb		
carrot	carrot	plant_organ		
	carrot	vegetable		
cauliflower	cauliflower	herb		
celery	celery	produce		
tomato	tomato	vascular_plant		
mushroom	mushroom	fungus		
	fungus	living_thing		
	mushroom	produce		
	mushroom	physical_phenomenon		
	physical_phenomenon	phenomenon		
	phenomenon	physical_entity		

object name	Concepts and Relations			
	is-a		HasA	MadeOf
pear	pear	produce		
garlic	garlic	ingredient		
beans	beans	herb		coffee
grapefruit	grapefruit	edible_fruit		
melon	melon	produce		
mango	mango	produce		
lime	lime	material		
	material	matter		
lemon	lemon	whole		
banana	banana	produce		
nectarine	nectarine	produce		
parsnip	parsnip	plant_organ		
	parsnip	vegetable		
ginger	ginger	ingredient		
	ginger	vascular_plant		
grape	grape	produce		
raspberries	raspberries	edible_fruit		
cucumber	cucumber	produce		
strawberry	strawberry	edible_fruit		
konjac	corm	plant_organ		
	plant_organ	physical_entity		

The meat and fish category name

object name	Concepts and Relations			
	is-a		HasA	MadeOf
cod	cod	sheath		
	sheath	natural_object		
	natural_object	physical_entity		
	cod	seafood		
	seafood	solid		
pork	pork	food		
chop	chop	meat		
	chop	natural_phenomenon		
	natural_phenomenon	process		
mince	mince	food		
prawn	prawn	food		
fishcakes	fish	region		
	region	physical_entity		
	fish	solid		
	fish	causal_agent		
	causal_agent	physical_entity		
mussel	mussel	seafood		
salmon	salmon	fish		
	salmon	stream		
	salmon	color		
	salmon	food		
steak	steak	meat		
sea bass	sea bass	fish		
	sea bass	seafood		
oyster	oyster	seafood		
fowl	fowl	food		
goose	goose	simpleton		
	simpleton	causal_agent		
	goose	bird		
	bird	food		
lamb	lamb	person		
	lamb	food		
squid	squid	food		

The perishable category name

object name	Concepts and Relations			
	is-a		HasA	MadeOf
cheddar	cheddar	dairy_product		
	dairy_product	food		
	cheddar	food		
	cheddar	settlement		
	settlement	region		
cheese	cheese	solid	flavor	milk
	cheese	foodstuff		
	solid	physical_entity		
	foodstuff	physical_entity		
egg	egg	food		
butter	butter	person		milk
	butter	solid		
	butter	foodstuff		
jam	jam	confiture		
	confiture	dainty		
	dainty	food		
	jam	gathering		
	gatering	event		
	gatering	group		
	event	physical_entity		
	group	physical_entity		
pastry	pastry	concoction		
	concoction	food		
	pastry	food		
buttermilk	buttermilk	fluid		
	buttermilk	part		
margarine	margarine	condiment		
mascarpone	mascarpone	cheese		
quiche	quiche	amerindian		
	amerindian	causal_agent		
pizza	pizza	nutriment		
coleslaw	coleslaw	dish		

object name	Concepts and Relations			
	is-a		HasA	MadeOf
smoothies	smoothies	deceiver		
	deceiver	bad_person		
	bad_person	causal_agent		
juice	juice	food		
	juice	electrical_phenomenon		
	juice	body_substance		
clementine	clementine	citrus		
	citrus	produce		
tteokbokki	rice	grass		
	rice	foodstuff		
chikuwa	skin	artifact		
	skin	causal_agent		
okonomiyaki	hiroshima	municipality		
	municipality	geographical_area		
	geographical_area	location		
	municipality	district		
	hiroshima	geographic_point		
	geographic_point	location		
nabemono	article	whole		
pfefferpotthast	span	digit		
	digit	number		
	number	whole		
	span	artifact		
spanferkel	pig	unpleasant_person		
	unpleasant_person	person		
	pig	block		
	block	whole		
	pig	lawman		
	lawman	preserver		
	preserver	artifact		
	pig	container		
fajita	container	artifact		

The product category name

object name	Concepts and Relations			
	is-a		HasA	MadeOf
glove	glove	clothing		
	clothing	artifact		
shampoo	shampoo	formulation		
	formulation	chemical		
soap	soap	club_drug		
	club_drug	drug		
	soap	formulation		
lotion	lotion	matter		
spray	spray	chemical		
	spray	vapor		
	vapor	mixture		
	mixture	matter		
	spray	decoration		
	decoration	whole		
	spray	container		
	spray	discharge		
	discharge	substance		
dishwasher	dishwasher	workman		
	workman	worker		
	worker	causal_agent		
tissue	tissue	material		
	tissue	part		
liners	liners	piece		
	piece	physical_entity		
rack	rack	locomotion		
	locomotion	change		
	change	event		
	event	physical_entity		
	rack	meat		
	rack	ending		
	ending	event		
	rack	supporting_structure		
	supporting_structure	artifact		
	rack	device		
	device	artifact		

object name	Concepts and Relations			
	is-a		HasA	MadeOf
brush	brush	process		
	process	part		
toothbrush	toothbrush	implement		
	implement	artifact		
roller	roller	bird		
	bird	chordate		
	chordate	organism		
	organism	whole		
	roller	solid		
	roller	pigeon		
	pigeon	gallinaceous_bird		
	gallinaceous_bird	vertebrate		
	vertebrate	animal		
	animal	living_thing		
duster	duster	whole		
plunger	plunger	person		
toilet roll	toilet roll	tissue		
bin	bin	artifact		
conditioner	conditioner	chemical		
cigars	cigars	tobacco		
	tobacco	drug		
	tobacco	plant_material		
lighter	lighter	substance		
candle	candle	device		
bulb	bulb	body_part		
	body_part	physical_entity		
	bulb	object		
	object	physical_entity		
toothpaste	toothpaste	cleansing_agent		
	cleansing_agent	compound		
	compound	material		

The medicine category name

object name	Concepts and Relations			
	is-a		HasA	MadeOf
cough syrup	cough	artifact		
pastilles	pastilles	candy		
	candy	sweet		
syrup	syrup	flavorer		
plasters	plasters	artifact		
	plasters	building_material		
	building_material	whole		
	plasters	substance		
bandage	bandage	cloth_covering		
	cloth_covering	covering		
	covering	artifact		
ointment	ointment	matter		
lozenges	lozenges	medicine		
	medicine	agent		
	lozenges	sweet		
	sweet	nutriment		
inhalator	inhalator	container		
	inhalator	device		
nicorette	auto	artifact		
senokot	senna	woody_plant		
	woody_plant	plant		
windsetlers	storage	facility		
	facility	whole		
	storage	possession		
	possession	activity		
	storage	operation		
	operation	physical_entity		
	storage	buildup		
	buildup	increase		
	increase	physical_entity		
	storage	device		
	storage	component		
	coponent	physical_entity		
revitalens	lense	device		
germolene	state	physical_entity		
nicolites cartomiser menthol	cigarette	tobacco		
decongestant	drug	agent		
linctus	medicine	agent		

The kitchenware category name

object name	Concepts and Relations			
	is-a		HasA	MadeOf
storage	storage	facility		
	storage	possession		
	storage	operation		
	storage	buildup		
	storage	device		
	storage	component		
jug	jug	vessel		
	vessel	container		
peeler	peeler	person		
	person	physical_entity		
	peeler	entertainer		
	entertainer	causal_agent		
poacher	poacher	acquirer		
	acquirer	causal_agent		
	poacher	container		
	poacher	kitchen_utensil		
opener	opener	causal_agent		
spoon	spoon	tableware		
	tableware	article		
baking tray	baking tray	kitchen_utensil		
funnel	funnel	chimney		
	chimney	conduit		
	conduit	way		
	way	whole		
	funnel	implement		
fork	fork	implement		
	fork	space		
	space	object		
spatula	spatula	cooking_utensil		
	cooking_utensil	kitchen_utensil		
mug	mug	person		
dish	dish	adult		
	adult	organism		
	dish	tableware		
	dish	food		

object name	Concepts and Relations			
	is-a		HasA	MadeOf
plate	plate	cut		
	cut	food		
	plate	course		
	course	location		
	course	food		
	plate	body_part		
	plate	tableware		
	plate	layer		
bowl	bowl	artifact		
	bowl	solid		
shaker	shaker	person		
	shaker	causal_agent		
teaspoon	teaspoon	container		
	teaspoon	cutlery		
	cutlery	ware		
	ware	artifact		
saucepan	saucepan	kitchen_utensil		
pan	pan	kitchen_utensil		
glass	glass	drug		
	glass	matter		
	glass		water	
tumbler	tumbler	athlete		
	athlete	person		
	tumbler	container		
coasters	coasters	traveler		
	traveler	causal_agent		
	coasters	inhabitant		
muffin tin	tin	chemical_element		
	chemical_element	matter		
hand mixer	mixer	food		
	mixer	equipment		
	equipment	artifact		
	mixer	liquid		
cake tester	cake	artifact		
	cake	food		
	cake	nutriment		

object name	Concepts and Relations			
	is-a		HasA	MadeOf
cafetiere	block	group		
	block	region		
	block	copy		
	block	whole		
saltball	pan	kitchen_utensil		
pot	pot	fixture	water	
	fixture	whole		
	pot	resistor		
	resister	device		
	pot	soft_drug		
	pot	kitchen_utensil		
cooker	cooker	kitchen_utensil		
chopper	chopper	aircraft		
	aircraft	vehicle		
	vehicle	physical_entity		
	chopper	bone		
	bone	animal_tissue		
	animal_tissue	body_part		
	chopper	edge_tool		
	edge_tool	cutting_implement		
	cutting_implement	implement		
blender	blender	kitchen_utensil		
grinder	grinder	machine		
	machine	physical_entity		
	grinder	snack_food		
	snack_food	nutriment		
	grinder	bone		

object name	Concepts and Relations			
	is-a		HasA	MadeOf
steamer	steamer	kitchen_utensil		
	steamer	vessel		
	vessel	vehicle		
	steamer	shellfish		
	shellfish	food		
jar	jar	container		
fryer	fryer	poultry		
	poultry	bird		
	bird	food		
scoop	scoop	solid		
	scoop	hand_tool		
	hand_tool	implement		
	scoop	club_drug		
mortar	mortar	artifact		
nutribullet	container	artifact		
sandwich press	sandwich	dish		
corkscrew	corkscrew	opener		

Appendix B: PUBLICATIONS

Kanjaruek, S. and Li, D. (2014) ‘Data-Information Retrieval based Automated Ontology framework for Service Robots’, In *Proceeding of the 2014 International Conference on Manipulation, Manufacturing and Measurement on the Nanoscale (3M-NANO)*, pp. 90-94.

Kanjaruek, S., Li, D., Qiu, R. and Boonsim, N. (2015) ‘Automated ontology framework for service robots’, In *Proceeding of the 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 219-224.