REALIZING PERVASIVE COMPUTING VISION: A CONTEXT-AWARE MOBILE APPLICATION APPROACH

ZHU CENZHE

(B.Sc., Shanghai Jiao Tong University, China)

Supervised by

Associate Professor TAY Teng Tiow

A THESIS SUBMITTED FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2014

DECLARATION

I hereby declare that the thesis is my original work and it has been written by me in its entirety.

I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

ZHU Cenzhe

February 26, 2014

Acknowledgments

The PhD years have shaped my thoughts about life and therefore I am glad that I took the decision to pursue graduate studies. The PhD journey has been one of the most challenging and rewarding journeys of my life. Hence, there are several people I would like to thank for helping me in this journey.

First and foremost, I would like to thank my supervisor, Dr. Tay Teng Tiow (Associate Professor, Department of Electrical & Computer Engineering, National University of Singapore). His patience and encouragement carried me on through all the difficult times, his insights and suggestions helped me to shape my research skills, and his valuable feedback contributed greatly to my research work. Prof. Tay has devoted much to help me to learn useful techniques and share his experience in academic research. I deeply appreciate his advice upon my research during these years.

I would like to express my appreciation to Prof. Wong Wai-Choong, Lawrence, for funding me with the POEM project. These muffled the distraction of financial concerns to a very great extent.

I am grateful to my PhD thesis committee members Prof. Bharadwaj Veeravalli and Prof. Akash Kumar for providing their valuable inputs to improve the thesis. I thank the Department of Electrical & Computer Engineering at NUS for supporting me throughout the program. This journey would not have been possible but for the collaboration with some wonderful colleagues. I therefore thank Chen Guo for helping me in the publications that we jointly published. I

would like to take this opportunity to thank the lab officers Mr. Ng and Ms. Ho in Digital Systems and Applications Laboratory. I wouldn't be able to concentrate on research without your help. I still remember the times we sit together and chat along to relax my tensed nerves.

I would also like to thank all colleagues in NUS IDMI Ambient Intelligence Lab and Information Systems Research Lab, Mr. Song Xianlin, Ms. Guo Jie, Mr. Fang Fang, Mr. Bao Yang, and many other good friends. Without you guys to have fun with, I cannot complete my thesis work and my PhD journey.

Last but not least, I want to dedicate this dissertation to my mother Mrs. Songhua Chen for her unselfish love, high moral support and encouragement to make me believe in myself to successfully complete all endeavour of my life so far. I would also like to thank my father Mr. Xianqi Zhu for the strength and wisdom he has given me to be sincere in my work and to become the better human being to contribute to well-being of the world. Most importantly, my very special thanks and love go to my fiancée, Jin Wang, whose stability, patience and loving care make for the right conditions so necessary for me to think deeply into research problems. And also, never underestimate the power of your encouragement.

February 26, 2014

Table of Contents

Summary

The vision of pervasive computing inspired the work of context-aware systems. Now, at a time where smart phones are ubiquitously available, development of context-aware mobile applications can drastically change how people think and how people live. With the goal of improving people's lives, this thesis works towards several problems in building context-aware mobile applications, specifically in building ontology-based context-aware mobile applications.

Firstly, in order to accelerate researches in this area, an ontology-based test bench is constructed. This test bench enables a shorter research cycle because new research ideas can now be tested on this test bench, saving the effort to build a real system. In addition, this test bench covers the domain of context-awareness extracted from hundreds of real mobile applications. Therefore, performance measured in this test bench can be more reliable than the one measured in existing benchmarks.

Secondly, one major problem in adopting ontology-based context model is the slow reasoning speed that hinders real-time deployments. Server cloning, which is a usual method to solve scalability problem, is not applicable for ontology databases due to the huge size of database and excessive synchronization traffic. In this thesis, a completeness-proven partitioning algorithm is proposed to enable a distributed computing scheme. In this scheme, sub-databases are extracted from the central database given the category of queries it is responsible of answering. The sub-database is extracted in a careful way so that the yielded

sub-database is "just-enough" to handle all queries of that category. This scheme significantly reduces the size of sub-databases. It also drastically improves the processing speed when an update if fired. The amount of synchronization traffic is minimized as well. While this result is by itself acceptable already, we can further reduce the cost by applying a trade-off, exchanging some overshooting completeness for the benefit of smaller sub-databases.

Last but not least, this thesis works out a context-aware recommendation algorithm that is applicable to context-aware mobile commerce applications. Context information, after being captured by a sensor or a crawler, is represented as a triple in the knowledge base. This triple is then quantified into a scale from 1 to 5, and it is plugged in the rating matrix. Following this, a modified collaborative filtering algorithm with weighting scheme is adopted to take the context information into account when making recommendations. The algorithm is tested in a movie recommendation scenario. Experiment results show our approach can decrease MAE and produce higher precision and recall. A prototype system on the domain of music recommendation is constructed and multiple users are invited to try and comment on it. The feedback from users shows the system is promising and it gives them positive mobile commerce experiences.

List of Tables

List of Figures

List of Symbols

v	A variable in First Order Logic
t, t_1, t_2	Terms in FOL
ϕ, ψ	Formulas in FOL
$\top$	All concepts in Description Logic
$\bot$	Empty concept in DL
R	A role in DL
C, D	Concepts in DL
a, b	Individuals in DL
$\mathbb{C}$	A set of classes (that denotes the domain of discourse)
$\mathbb{P}$	A set of properties (that denotes the domain of discourse)
C_i	An OWL class
P_i	An OWL property
$\mathbb{NC}$	A set of classes. It denotes a filter which passes only individuals that are explicitly NOT of the classes in the set.
$\mathbb{E}$	A set of individuals. It denotes a filter which passes triples that have either side being the individuals in the set.
N_C	The number of classes in an ontology
N_P	The number of properties in an ontology

N_T	The number of triples in an ontology
$\mathbb{K}$	The set of all triples in a knowledge base
$\mathbb{S}$	The set of triples that are included in the sub-database
f_i	The ith element of the fact sequence in inference tree
a_i	The ith element of the axiom sequence in inference tree

List of Abbreviations

PC Personal Computer

M-commerce Mobile Commerce

MFS Mobile Financial Services

UML Unified Modeling Language

FOL First Order Logic

DL Description Logic

TBox Terminology Box

ABox Assertion Box

OWL Web Ontology Language

RDF Resource Description Framework

RDFS Resource Description Framework Schema

TTL Terse RDF Triple Language

SPARQL SPARQL Protocol and RDF Query Language

SDK Software Development Kit

RPC Remote Procedure Call

IPC Interprocess Communication

WP Windows Phone

UI User Interface

CC/PP Composite Capability/Preference Profiles

CSCP Comprehensive Structured Context Profiles

ORM Object-Role Model

ER Entity-Relationship

IDL Interface Definition Language

CF Collaborative Filtering

GPS Global Positioning System

UID Unique Identifier

CCF Context-aware Collaborative Filtering

Chapter 1

Introduction

1.1 The History and Current State

Two decades ago, Mark Weiser[1], the harbinger of Pervasive Computing, envisioned a highly intelligent world where computing resources are so ubiquitous that they fade away from people's focus. The "live board" that was advocated in the paper very much resembles products that we have 20 years later—iPad^{TM}and other tablet devices. Numerous researchers are inspired by the vision and we do have seen great advances in realizing this vision. But have we reached there yet? Or, perhaps the vision is so ahead-of-time that we actually have just reached the starting line?

I prefer the latter answer. That is, there is still a long way to go before we reach our goal, and the most exciting part has just come in. Technology revolution happens only when the infrastructure is set up and people's minds are ready. Twenty years ago few people have access to Personal Computers (PCs),

not to mention weaving the technology into the background. Some of the very novel ideas implemented for pervasive computing, like the Active Badge[2] and the PARCTAB[3], failed to get commercialized or generalized partially because of their user intrusiveness. You cannot expect users to carry around a palm-size gadget that has only one functionality of tracking themselves. While at the same time we cannot integrate many functionalities into one gadget due to the limitation of computation power. These years have seen the proliferation of PC, the exponential computing speed boost, and recently the emergence of smart phones. The computational power of computers and devices has reached a level that is sufficient to embody a decent amount of intelligence. Smart phones and mobile data networks have become almost ubiquitously available, and this enables a series of scenarios of pervasive computing. According to a survey in 2012[1], there are a total of 1.08 billion smart phone users globally, and Singapore has the highest smart phone penetration rate in the world of 54%. The recent Google Glass™ and Apple iWatch™ further augment the varieties of unobtrusive intelligent computation media. A new round of revolution of pervasive computing is now ready to launch, starting from the revolution in mobile applications.

The hardware infrastructure agrees with pervasive computing, the next question is whether people's minds are ready for the change. The fact is, people are looking forward to the change and they are already practicing the change of life style. According to the same survey as last paragraph, 89% of smart phone

[1] http://www.go-gulf.com/blog/smartphone/

users use their phones throughout the day, and the amount of Internet data usage reach as high as 582 MB a month per capita. Smart phones have integrated themselves into our lives and users even cannot live without them. The use of smart phone is no longer limited to calling, SMS, or browsing web pages. Mobile Commerce (M-commerce) emerges as a result of user payment habit shift. This includes near-field payments, M-ticketing, M-coupons, M-banking, M-wallets, remittances and other Mobile Financial Services (MFS). According to IDC Financial Insights 2012 Consumer Payments Survey, 34 percent of survey respondents have made a purchase using their mobile phone compared to 19 percent in 2011. The report also found that physical goods were the most common mobile purchase, with more than 70 percent having purchased a physical good. 60 percent have purchased online services and digital goods instead. Japan, being the king of M-commerce, even has forecasted US$119 billion revenue in 2015. This is about 8% of the total E-commerce market.

In such a background, now is the best time ever to promote the development and deployment of pervasive computing techniques. And we start with the context-aware mobile application approach. Context-aware systems and applications were initially designed to realize Weiser's vision. Context-aware application refers to an application that is able to detect the context of its user, and to tune its behaviour according to the context, and further make an impact on the user's behaviour[4]. This is significantly different from the most of the popular mobile applications we have on smart phones. While most of the current mobile applications are merely a portable edition of the applications on

stationary computers, context-aware mobile applications exploit the advantages of smart phones that they are closer to the users and they are able to sense the context of the users. By Dey[5, 6]'s definition, context is any information that can be used to characterize the situation of an entity. This usually includes the surrounding environment, the personal profile, and the preference settings of the user. Context-aware systems, however, refer to the middleware that standardizes and integrates different parts of context-aware applications. The effort is made so that context-aware application developers can concentrate on the core logic or business model instead of the low-level sensor data manipulation.

The thesis works on the domain of context-aware mobile applications. Specifically, the work conducted can be divided into three parts. The first part solves the problem of the lack of experimental test bench in this domain. The second solves a problem in scaling up context-aware mobile systems by introducing a distributed computing scheme. The third part proposes a context-aware recommendation system that is of great importance in mobile commerce applications. The motivation of these works is described in details below.

There are many research directions in the domain of context-aware systems, among which the most basic one is how to represent and store context information[7]. The method used for the knowledge representation and storage in a machine processable form is called *context model*. Among various context models, *ontology-based model* for context-aware systems has its strength in distributed composition[8], strict semantics, the ability to be verified and reasoned and many more[9, 10]. Other models include key-value pair models[11, 12, 13],

mark-up scheme models[14, 15, 16], graphical models stored as UML (Unified Modelling Language)[17, 18], and object-oriented models[19].

For the above mentioned reasons, ontology-based model is chosen as the context model in our system and it is used throughout the thesis.

Though the ontology-based model has many advantages, research activities are usually thwarted by the lack of an ontology-based test bench on the domain of context-aware systems. This is because research ideas in this domain are usually highly data-dependent, and the performance measured in existing ontology-based benchmarks would be unreliable. Researchers would have no choice but to actually build one whole system in order to test out their ideas. Seeing this, this thesis works on building a test bench specifically on this domain from scratch. Initially, a survey of hundreds of mobile applications is done. Current mobile applications are quite similar to the concept of context-aware applications. Or rather, some of the applications are already context-aware, to some extent. By modelling the query types and data structures of these applications, we can extract fractions of the whole knowledge base. And these pieces are finally integrated together to form the upper-level ontology in the domain of context-aware systems. This upper-level ontology, together with other important components, constitutes the test bench.

With the test bench ready, this thesis then works towards the scalability issue in ontology-based context-aware systems. The major shortcoming of ontology-based context-aware systems is that the ontology processor (aka. ontology reasoner) is relatively slow for real-time requirements. Thus, we are facing the scal-

ability problem when the number of users grows beyond the capability of one central server. This thesis proposes a distributed computing scheme for ontology-based context-aware systems. Under this scheme, monolithic ontology knowledge bases are carefully examined and partitioned so that the query-answering task can be distributed among a number of servers. This can greatly enhance the processing speed, thus promoting the usage of ontologies in real-time context-aware systems.

Mobile commerce emerges around 2000[20] and is now a pivotal component in the domain of mobile applications. M-commerce is a subset of E-commerce and is usually defined as "any transaction with monetary value that is conducted via a mobile network"[21]. Back in 2009, Chang [22] surveys the features and characteristics of contemporary popular smart phones, putting an emphasis on the required and desirable features for mobile commerce. Though the smart phone market grows with unprecedented speed past the release of this paper, several points of this paper are proved to be quite insightful and predictive. Currently, there is an emerging trend that many banks and financial institutions are making their moves to provide their services on users' smart phones. For example, Standard Chartered Bank provides *Breeze*[2] to simplify personal banking procedures. Chase introduces the mobile application *Chase My New Home* to help home-buyers from the time they start looking at houses until they close on their mortgages[3]. And these mobile applications are starting to embrace context

[2]http://www.standardchartered.com.sg/personal-banking/online-mobile-services/breeze-mobile/en/

[3]https://www.chase.com/mortgage/home-loans

information in their development.

This thesis works specifically on context-aware recommendation systems. Recommendation systems adopted in E-commerce are proved to be of great importance. The recommendations are given from a rating matrix, which is an integration of all users' history and preferences. In M-commerce, the recommendations can be augmented with the extra information of the users' context. As such, a well-designed and specially-tuned context-aware M-commerce recommendation system plays a critical role in promoting the usage of M-commerce.

Collaborative filtering (CF) has been a successive solution for recommendation systems. Adding contextual information to collaborative filtering has also been actively studied for some time already. The basic context information is time. It is important to determine what to deliver to the customer as well as when. For example, one might prefer reading world news and stock market updates on weekdays, but prefers reading movie reviews and shopping catalogues. Location, budget, personal interest, friend collocation and many more contexts can be exploited to provide better recommendations. There can be some types of contexts that we are not even aware of their relevance to recommendation choices. With the inclusion of context information, we can significantly improve the accuracy of recommendations given.

On the domain of ontology-based context-aware systems and M-commerce, many accomplishments have been achieved. But we still face many challenges. Next I will formulate the challenges but leave the accomplishments in Chapter 3 Literature Review.

1.2 Key Research Challenges

When implementing and deploying ontology-based context-aware systems, we are facing the following challenges:

- Heterogeneous context information types. Context information can be sensor data extracted that have a numerical form. It can also have a form of a qualitative value. Things get more complicated with the introduction of partial orders, entailments, and conjunction / disjoints / mutually exclusive relationships. The design of an ontology that depicts all the details requires effort.

- Slow reasoning speed for real-time requirements. Over decades researchers have been working to build an efficient ontology reasoner. Implementing the tableaux algorithm described in [23], many state-of-the-art ontology reasoners are built, such as *RacerPro, FaCT++, Pellet* and *HermiT*. Though we are pleased to see the advances in reasoning speed, we have to admit that a single reasoner still cannot fulfil high-load real-time tasks[24].

- Convoluted ontology structure. The structure of an ontology is much more complicated than a table view in relational databases. The knowledge base is formed of a large sum of triples that are interwoven with each other. Therefore, isolating "useful" and "useless" triples from an ontology base can be difficult. It also makes the construction of a synthetic knowledge base more difficult because all entities should be determined before relationships are added to the graph.

On the domain of context-aware recommendation systems, the challenges we face can be summarized as:

- The introduction of context information invalidates common recommendation algorithms. Typical recommendation algorithms have the rating matrix as the sole input. Context information cannot be represented in such a framework.

- Existing context-aware recommendation systems have this and that problems. They typically introduce an extra dimension, then use certain filter rules to project the 3-D spaces to 2-D ones. However, the selection of the filter rules must be hand-picked, and that requires a lot of effort and it is subject to errors. In addition, projecting the 3-D rating table to a 2-D one is based on a yes-or-no filter. Thus, it loses the quantitative value of the context information.

- The sparsity problem, which is a common challenge for recommendation algorithms, also exists for context-aware recommendation systems. When the data set of user inputs is small, the quality of the recommendation algorithm can be severely degraded.

1.3 Contribution of the Thesis

The central idea of this thesis is to promote the development of context-aware systems. All three parts in the thesis work towards the same goal, while they are interconnected to each other. Specifically we have:

1. Constructed a test bench for ontology-based context-aware systems. The domain of context-aware systems is very different from legacy ontology benchmarks. While maintaining the convoluted nature of existing benchmarks, it has distinctive features. For example, ontology for context-awareness generally has shallower structure with several deep branches. This test bench includes an ontology that covers 20 different categories of context-aware applications, a great many synthetic concrete triples that are populated by a set of rules, additional triples to reflect distributed composition, and a set of queries to mimic the behaviour of context-aware applications.

2. Developed a completeness-proven algorithm to enable distributed computing scheme in ontology-based context-aware systems. Using the algorithm described, we can extract application-specific sub-databases from the whole knowledge base. It is also proved that the extracted sub-database can perfectly accommodate queries from that specific application, which is also known as the completeness of the sub-database (or algorithm). The algorithm also covers the synchronization process in distributed computing. When an update of information is received at the server, it can be quickly decided (without going through an ontology reasoning process) whether this update should be delivered to other servers.

3. Devised a new context-aware recommendation algorithm. This algorithm managed to represent context information as well as user ratings in 2-D

space, thus existing recommendation methodologies can be utilized with minimal modifications. In addition, the dimension reduction in our approach is not based on a yes-or-no filter like previous works, it keeps the quantitative information attached to context information, thus provides richer data as well as higher precision.

The first part of the work, i.e. the test bench, serves as the glue to consolidate the works. The distributed computing scheme for ontology-reasoning is evaluated both on our own test bench and on other existing benchmarks. The mobile application extends our test bench (an upper ontology) with expert knowledge on music (domain-specific ontology).

1.4 Thesis Organization

Chapter 2 summarizes some preliminaries and the nomenclatures of *First Order Logic*, *Description Logic* and *Ontology*. These are especially important in comprehending the algorithm described in Chapter 5.

Chapter 3 gives a thorough literature review on the field this thesis is concerned, specifically, context-aware systems, semantic web applications, M-commerce and recommendation systems.

Chapter 4 first explains the motivation of building a test bench in the domain of context-aware applications. Several existing ontology-based test benches are then studied and their limitations are exposed. It is then followed by the overall description and the details of the construction of the ontology-based test bench

on the domain of context-awareness. And finally, a series of comparisons between our test bench and existing ones are done to complete this work.

In Chapter 5, we proposed a fast and complete algorithm to extract sub-ontologies from a base ontology for a given task, and also to keep the sub-ontology updated whenever changes are issued to the base ontology. With this algorithm, a distributed computing scheme is then applicable to the ontology-based context-aware system. This scheme is achieved so that the processing speed of queries and updates can be improved thousands of times, while the cost of the scheme is usually constrained to tens of times of storage cost. After that, a tuning process can be applied to further tune the performance of the algorithm. This is essentially balancing a trade-off between excellent completeness and smaller storage cost. Investigation reveals that most of the time we can achieve certain amount of savings with no direct impact to the query-answering quality, while further reducing storage cost can harm the query-answering accuracy.

Chapter 6 focuses on our works on context-aware recommendation systems. In this chapter, we proposed a novel recommendation system that is able to utilize context information. Context information, after being captured is quantified and plugged into the rating matrix. A novel recommendation algorithm that overcomes the shortcoming of existing systems is proposed. It managed to process context information within 2-D space, while fully maintaining the quantitative value of context information. This change rid us from setting an arbitrary numerical threshold or a cut-off qualitative context, and we can benefit from the quantitative effect of context information, which finally leads to a

higher recommendation precision.

And finally in Chapter 7, we conclude this thesis by summarizing all the works.

Chapter 2

Preliminaries

2.1 First Order Logic

First order logic (FOL) is a formal system used in mathematics[25]. It is also known as predicate logic. We introduce some preliminaries for first order logic here because all of the logic concepts covered in this thesis are under the domain of first order logic. All data models, including Description Logic and Ontology are extensions of FOL.

FOL requires the parameter to its predicate to be only variables (no other predicates or more quantifiers). The basic building block of FOL is variables (like a) and functions of variables (like $f(a_1, \ldots, a_n)$), and this building block is called as *term*. *Terms* can be combined with other elements to form *formulas*. A *formula* is an expression in FOL that maps each possible variable value to a truth value. The extension of a formula is the set of variable values that would be mapped to *TRUE*. The composition of a formula is defined recursively. Suppose

v is a variable, t, t_1, t_2 are terms, ϕ and ψ are formulas, the following are all formulas:

1. $P(t)$ where $P(.)$ is a predicate. Predicate is the most basic formula that represents a meaning.

2. $t_1 = t_2$. Equality can be considered as a special predicate that equates two terms.

3. $\neg\phi$. Any formula of FOL can be negated using the negation symbol.

4. $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \rightarrow \psi$ are all formulas. Binary connectives of formulas are also formulas.

5. $\forall v.\phi$ and $\exists v.\phi$ are both formulas. $\forall$ denotes universal restriction and $\exists$ denotes a existential restriction.

Each formula states a piece of information. Sometimes a number of pieces of information can be combined together to derive new formulas. This is called deductive reasoning. For example, suppose we have 2.1 stating that Socrates is a philosopher. Adding in the knowledge stated in 2.2 that all philosophers are mortal, we can derive 2.3 that Socrates is mortal.

$$\text{Philosopher(Socrates)} \tag{2.1}$$

$$\forall a.(\text{Philosopher}(a) \rightarrow \text{Mortal}(a)) \tag{2.2}$$

$$\text{Mortal(Socrates)} \tag{2.3}$$

Table 2.1: Description Logic and First Order Logic Equivalences

DL	FOL
concept	class
role	property
individual	object

FOL has the following notions. These notions would be compared to the ones in Description Logic and Ontology later:

object A specific value of a variable. For example, *Socrates*.

property A predicate. For example, *Mortal* and *Philosopher*.

class A set of objects. For example, all the philosophers.

2.2 Description Logic

Description Logic (DL) [23, 26] is a family of formal knowledge representation languages. In fact, it is a sub-set of FOL. A Description Logic models concepts, roles, individuals and their relationships.

In DL, a database is called a knowledge base. It can be divided into TBoxes and ABoxes. TBox, short for Terminology Box, is a set of assertions that defines the syntax of a language. These assertions are also called axioms. Specifically, declaration of classes and properties constitute the TBox. TBox is also referred to as the vocabulary of a knowledge base. ABox (Assertion Box) is the part of a knowledge base other than the TBox. Assertions in ABox denote concrete information that is written following the vocabulary of the language (TBox). For example: In the TBox of a knowledge base, we defined classes (*People, Location*)

Table 2.2: Description Logic Symbol Descriptions

Symbol	Description
$\top$	All concepts. A concept that includes all individuals.
$\bot$	Empty concept. A concept that has no individual.
$\sqcap$	Intersection/conjunction of concepts.
$\sqcup$	Union/disjunction of concepts.
$\neg$	Negation/complement of concepts.
$\forall R.C$	Universal restriction. It means a concept whose individuals all have the role of R and have the object of C.
$\exists R.C$	Existential restriction. It means a concept such that some of its individuals have the role of R and the object of C.
$C \sqsubset D$	Concept inclusion. It means all C concepts are D concepts.
$\equiv$	Concept equivalence.
$a : C$	Concept assertion. Individual a is a C.
$(a, b) : R$	Role assertion. It means a is R-related to b.

and property (*LocatedIn*). Then we are able to include assertions like <*Alice, is, Girl*>, <*Singapore, is, Location*>, and <*Alice, LocatedIn, Singapore*> in the ABox of the knowledge base.

Some special symbols and expressions used in DL are listed in Table 2.2

Description Logic has many dialects, depending on their expressiveness. Please refer to Table 2.3 for the naming convention. These dialect symbols each denotes a type of expression allowed in the dialect. Combining these symbols can produce home-made Description Logic dialects. Among all of the DL dialects, three of them are most often used. They are $\mathcal{ALC}$, $\mathcal{SHIF}(D)$, and $\mathcal{SHOIN}(D)$.

2.3 Ontology

An ontology formally represents knowledge as a set of concepts within a domain, and the relationships between pairs of concepts. It can be used to model a domain and support reasoning (knowledge derivation) about concepts.

Table 2.3: Description Logic Expressiveness Naming Convention

Dialect Symbol	Description
$\mathcal{AL}$	Attributive language. It allows atomic concept negation, concept intersection, universal restriction and limited existential quantification.
$\mathcal{FL}$	Frame-based language. It allows concept intersection, universal restriction, limited existential quantification and role restriction.
$\mathcal{EL}$	This allows concept intersection and full existential restriction.
$\mathcal{F}$	Functional property.
$\mathcal{E}$	Full existential qualification.
$\mathcal{U}$	Concept union.
$\mathcal{C}$	Complex concept negation.
$\mathcal{H}$	Role hierarchy.
$\mathcal{R}$	Limited complex role inclusion axioms; reflexivity and irreflexivity; role disjointness.
$\mathcal{O}$	Classes with enumerated objects.
$\mathcal{I}$	Inverse property.
$\mathcal{N}$	Cardinality restriction.
$\mathcal{Q}$	Qualified cardinality restriction.
(D)	Use of datatype properties, data values, or data types.
$\mathcal{S}$	An abbreviation of $\mathcal{ALC}$ with transitive roles.

OWL (Web Ontology Language)[27] language is frequently used to model ontologies. OWL uses Resource Description Framework (RDF)[28, 29] as its internal structure. In RDF, facts are denoted as triples consisting of a subject, predicate and an object. For example, new classes and properties are defined by putting the name of the class or property as subject, *rdf:type* as predicate, and *rdf:Class* or *rdf:Property* as object. RDFS (RDF Schema)[30] extends RDF by providing mechanisms for describing groups of related resources and the relationships between these resources. RDF Schema vocabulary descriptions are written in RDF using the terms described in this document. These resources are used to determine characteristics of other resources, such as the domains and ranges of properties. OWL extends RDF by adding a set of constraints and new

vocabularies to it. In OWL, we can enforce more restrictions on properties like *owl:allValuesFrom*, *owl:someValuesFrom*, etc.; we can also declare properties to have new characteristics like transitivity, symmetry, or declare one property to be inverse to another one; we are provided with mechanisms to enforce equality, inequality and cardinality constraints as well.

Following is a brief listing of the terms we use in ontologies and OWL:

- Triple. Triple is the basic building block for RDF and OWL. It consists of a subject, a predicate and an object. All facts can be abstracted as triples just like we can use English sentences to denote information. Sometimes triples are referred to as assertions because a triple asserts a fact.

- Individual. An object or an instance. These are the most basic objects.

- Class. Class is an abstraction of a kind of individuals. In OWL, classes can be used as subject or object in a triple. Individuals that belong to a class can be used as subject or object of a triple as well.

- Property/Attributes. Property is a special type of predicate in triples. It is used to manifest a property of an individual, either a relationship to another individual, or an attribute of the stated individual. The domain of a property is a class such that individuals that have this property are instances of this class. Similarly, the range of a property is the class whose individuals can be the object of the property. In OWL, there are mainly 2 types of properties. ObjectProperty is used to denote relationships between individuals, while DataProperty is used to attach a typed literal to the

subject individual.

- Restrictions. Formally stated descriptions of what must be true in order for some assertion to be accepted as input. They are usually used in the definition of classes.

- Axioms. Axioms are similar to the concept of TBox in Description Logic. Axioms define classes and properties, and thus provide a vocabulary for concrete information representation.

When an ontology is given, specific procedures should be undertaken to check the consistency of the ontology. Most of the time, we may also want to derive new knowledge from the existing ones, i.e., do reasoning on the ontology. These are aided by Tableaux algorithms[31]. Tableaux algorithms are the currently mostly used algorithms for ontology consistency checking and reasoning. Many researchers have devoted themselves in building ontology reasoners with different optimization techniques. These include Jena[32], Racer[33], FaCT++[34, 35], Minerva[36], Hermit[37] and many more. However, tableaux-based decision procedure for the consistency of general $\mathcal{ALC}$ knowledge base runs in worst-case non-deterministic double exponential time.

OWL2[38] is a successor work to OWL language. It introduces new features: a. Syntactic sugar, b. New constructs for properties, c. Extended datatype capabilities, d. Simple metamodelling capabilities, e. Extended annotation, etc. A simple versioning technique is introduced also.

2.3.1 Turtle Language

OWL, being a theoretical model for ontologies, does not directly specify the serialization method. An XML-based serialization method is usually adopted to store ontologies. This method is usually referred to as RDF/XML. However, this method produces way too much expressiveness than OWL requires. Therefore, the output file produced often contains too much redundancy. In order to reduce this cost, Turtle (TTL, or Terse RDF Triple Language) is proposed to remove the undesirably prolonged XML syntax. Fig 2.1 shows the representation form of RDF/XML and TTL for the same property. Disregarding the imports and prefixes, it is obvious that TTL produces a much more succinct presentation of knowledge. In our works, we use TTL for all ontology serializations.

2.3.2 SPARQL

SPARQL (SPARQL Protocol and RDF Query Language) is an RDF query language, that is, a query language for databases, able to retrieve and manipulate data stored in Resource Description Framework format. [39] defines the syntax and semantics of the SPARQL query language for RDF. SPARQL is a language designed to formally express queries upon diverse RDF data sources despite of the native storage methods. Results of SPARQL queries can be result sets or RDF graphs. The *SELECT* query returns variable bindings; the *CONSTRUCT* query returns new RDF graphs; the *ASK* query returns a Boolean value indicating whether a pattern is found or not; and a *DESCRIBE* query returns an RDF graph that describes the resource found. Variables are denoted with a initial

```
<rdf:RDF xmlns="http://example.com/ontology#"
  xml:base="http://example.com/ontology"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

  <owl:ObjectProperty rdf:about="http://example.com/ontology#CoachOf">
    <rdfs:domain rdf:resource="http://example.com/ontology#SportTeam"/>
    <rdfs:range rdf:resource="http://example.com/ontology#Person"/>
  </owl:ObjectProperty>
</rdf:RDF>
```

Above is in RDF/XML format.

```
@prefix owl:  <http://www.w3.org/2002/07/owl#> .
@prefix :  <http://example.com/ontology#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@base <http://example.com/ontology> .

:CoachOf rdf:type owl:ObjectProperty ;
   rdfs:range :Person ;
   rdfs:domain :SportTeam .
```

Above is in TTL format.

Figure 2.1: RDF/XML and TTL representing the same property

```
PREFIX dc:  <http://purl.org/dc/elements/1.1/>
PREFIX ns:  <http://example.org/ns#>
SELECT ?title ?price
WHERE  ?x ns:price ?price .
   FILTER (?price < 30.5)
   ?x dc:title ?title .
```

Figure 2.2: SPARQL Query Example

question mark, e.g., ?x. We can also define restrictions on the pattern, or make projection and sorting procedure over query results. Queries that involve more than 1 RDF graph are supported as well. SPARQL adopts a syntax similar to TTL as shown in Fig 2.2

SPARQL Update[40], aka. SPARUL, is an extension of the update functionality to SPARQL language. Update operations are performed on a collection of graphs in Graph Store. SPARUL is currently NOT a standard yet. Jena supports SPARUL.

2.4 Chapter Summary

This chapter mainly refreshes the background on First Order Logic, Description Logic and Ontology. In the following chapters, these materials can be of help in understanding some of the abbreviations and terminologies.

Chapter 3

Literature Review

3.1 Existing Mobile Application Platforms

Smart phones are the ideal and practical media for the vision of pervasive computing. Their ubiquitous availability, computational power, and wireless communication capacity enable them to host context-aware applications. However, current mobile application platforms are not yet ready for incubating context-aware applications of higher complexity. In this section, we discuss the features of several existing mobile application platforms and demonstrate why a middleware for context-awareness is required.

Google released *Android*[41, 42] in November 2007 with the goal of being an open source arena for software development on mobile platform. At the time of writing, the newest release of Android Software Development Kit (SDK) is revision 17 for Android 4.2. Android is an open source mobile operating system based on the Linux kernel. The operating system facilitates developers

to write managed code in Java using Google developed Java libraries. Mobile applications written for Android are run on a Java virtual machine named Dalvik, which is specially designed for Android by Google. When developing context-aware mobile applications on Android system, developers could use android.hardware.SensorManager to read sensor data and subscribe to sensor events. In other words, the Android platform itself does not provide any sensor aggregation support. Developers will need to cope with raw sensor data directly, which is undesirable. Android uses remote procedure calls (RPC) as a mechanism for interprocess communications (IPC). This mechanism enables applications to share information with other applications. But this tightly-coupled communication between processes severely restricts the scope of data sharing. Developers are looking for a more loosely-coupled sharing mechanism between applications.

Another popular mobile application platform is *iOS*. iOS is initially designed for *iPhone* in 2007, and it has evolved ever since then. At the time of writing, the newest version is iOS 7. iOS is derived from *OS X*, which is the operating system for Apple computers. Developers use Xcode on Mac OS to develop mobile applications for iOS[43]. The programming language used is Object-C. Similar to Android, iOS SDK does not support sensor data aggregation[44]. Though it allows developers to access accelerometer, gyroscope, GPS, proximity and other sensor data, it is difficult to manage various kinds of sensor data. The battery drains very fast if each application tries to directly access sensor readings.

Windows Phone (WP) is another common mobile application platform[45]. It

is developed by Microsoft and it is a successive work following *Windows Mobile*. The latest version is WP 8. Featuring a different user interface (UI), Windows Phone uses Visual Studio as the development environment. The language used in the development is C++/C#. This paper [46] compares several different mobile application platforms in multiple perspectives. Windows Phone supports multiple sensors that allow apps to determine the orientation and motion of the device. Windows Phone provides with a combined motion API that processes input from all sensors. This API can be seen as a type of sensor data aggregation. However, the way these sensor data are aggregated is fixed beforehand and it is not programmable by developers. Windows Phone also provides API for each single sensor data reading.

In order to simplify the development of context-aware mobile applications, a more sophisticated platform is called for. Alternatively, these platforms can be augmented by a context managing middleware. In the following section, we describe the common forms of context-aware systems.

3.2 Context-aware Systems

There have been numerous studies on the design, implementation, analysis and optimization of context-aware systems. Through the time-line from [10], [9], [47], to [7], their surveys on this area have summarized what we have learned from previous research, and what are left to be done.

In this subsection, we will demonstrate the different context modelling approaches recently used in context-aware systems in terms of their advantages

and disadvantages. We then proceeds to discuss the various context-aware system architectures that have been proposed, outlining their main features, the differences between these architectures and the approach proposed in this thesis, and the ways in which these architectures could be improved by our proposed distributed computing scheme.

"In order to better understand how we can use the context and facilitate the building of context-aware applications, we need to more fully understand what constitutes a context-aware application and what context is." By Dey[5]'s definition, context is any information that can be used to characterize the situation of an entity. The author also provides an initial set of primary context types—*location*, *identity*, *time* and *activity*. This is the pioneering work in the field of context-aware systems.

3.2.1 Context modelling Methods

Knowledge in context-aware systems requires a unified method of modelling and representation. The different types of clients and providers that need to understand each other require that context information be represented uniformly throughout the system. The system must also be flexible and extensible enough to handle the wide range of context types, as well as the relationships between them. Most context-aware systems are distributed; it is therefore necessary for context models to be easily shared especially in our architecture, given its use of profiles and views. Models should also have a high level of formality and be able to represent existing context relationships. For these reasons, researchers

have developed a number of different methods for modelling context information. Some of these models do not meet the requirements imposed by context-aware systems, while others can be used as the basis for our ontology. In the following text, we provide an overview of available context models.

CC/PP (Composite Capability/Preference Profiles)[14] is based on RDF and focuses mainly on describing capabilities and preferences for wireless devices and mobile phones. Indulska[48] extends the vocabulary of CC/PP from description of device/user profiles to basic classes of context information needed in the infrastructure of pervasive systems. However, the way in which CC/PP is structured makes it difficult to capture and represent all context information such as accuracy, resolution, as well as the temporal characteristics needed to model the freshness of acquired context. The Component-Attribute model used in CC/PP causes many difficulties for multi-layered attributes, and the process of profile creation. CC/PP does not have the ability to constrain the list of elements within a container for a certain cardinality or type. Finally, CC/PP does not support any relational constraints for attributes within a profile component.

[15] investigates CC/PP and IETF's media feature sets. It reveals that CC/PP fails to achieve some of the design goals. IETF's media feature sets are not decomposable and not extensible. Hence, the Comprehensive Structured Context Profiles (CSCP)[49] is raised. CSCP overcomes the shortcomings of CC/PP in structuring and extends the external references and defaults mechanism. CSCP also provides a mechanism extending user preferences to incorporate conditional and prioritized attributes. However, the use of RDF syntax

for profile representation makes it an unlikely candidate for context representation due to the existence of more powerful ontology languages such as RDFS and OWL. Secondly, CSCP is still limited by its vocabulary to a limited set of context concepts and relationships.

Henricksen et al.[17] introduced his ORM-based (Object-Role Model) graphical context model, in hoping of overcoming the lack of formality and expressiveness present in the available context modelling approaches. Existing graphical models, such as UML and ER (Entity-Relationship), have some difficulties in modelling all features associated with context information. These features include uncertainties, histories, and dependencies between different types of information. In [18], several issues on context-aware software engineering are studied. The authors proposed a set of conceptual models to address these issues.

Graphical models like ORM have a strong ability to describe structures of context knowledge and to derive the required code, such as the one associated with ORM's relational code. Unfortunately, merging distributed context models is not fully effective, given the constraints associated with the act of merging relational databases. Since graphical models are used mainly to facilitate human readability, most graphical-based context modelling approaches present difficulties in merging different context models and would not be suitable for resource-deficient mobile devices.

Jean Bacon et al.[50] proposed a rule-based context modelling method utilizing formal logic in their location-oriented multimedia system. They developed a system that used an event-based mechanism to support location awareness.

Bacon extended the Interface Definition Language (IDL) to handle the occurrence of events such that servers could declare the events they were capable of notifying.

Logic-based context models have a high level of formality and express context information in the form of facts, expressions and rules where context information facts are added, deleted or updated within the system. However, creating a complete set of rules capable of representing the wide range of concepts, relationships, and properties within context-aware environments is a complex and time consuming task, rendering the use of logic-based context models unfavorable.

Ontologies, as a promising means for knowledge representation and sharing, have gained recognition in multiple disciplines. This is mainly because ontologies can represent concepts and relationships by employing a computer-usable data structure while sharing a common understanding of the domains in which the context-aware system has interest. There are many survey papers on employing ontology model for context information representation, and here we only name a few[9, 47, 7]. Since this part has been researched extensively, we here only briefly explain it. Ontological knowledge is usually represented through a set of entities, functions, instances and axioms. Ontologies are also known for their normalization abilities and formality, making them a favorable candidate for modelling context knowledge. Most proposed ontology-based context models have adopted the OWL language as a means of ontology representation. This is due to OWL's superior expressive abilities over other languages, such as XML, RDF, RDFS, or DAML+OIL. OWL also allows the exchange and comprehension of context

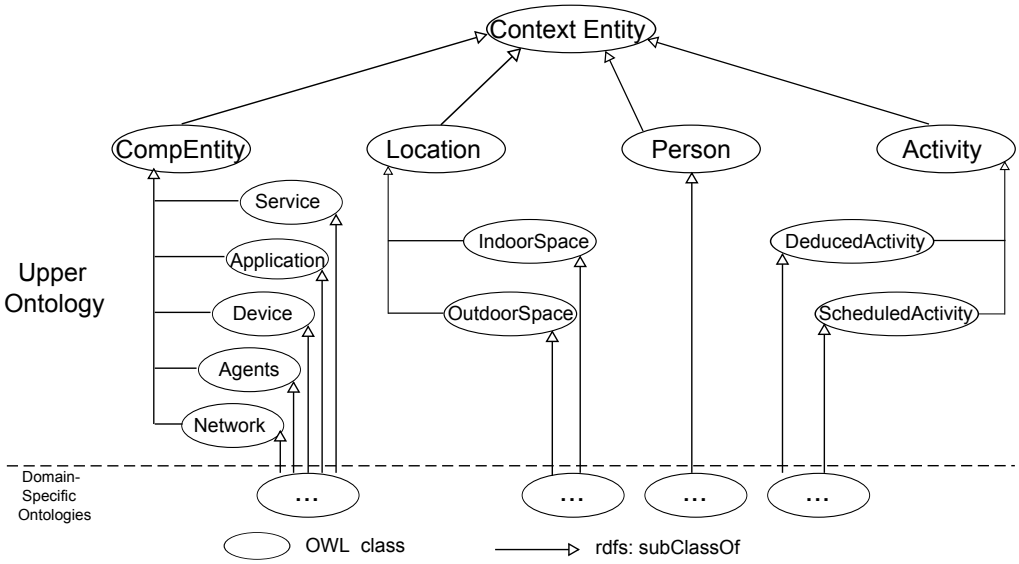

Figure 3.1: Upper-layer ontology of SOCAM

information between different entities within context-aware systems. The availability of a number of OWL-based reasoning engines that can interpret context information present within the ontology, or that can infer new context from existing context and relationships, makes OWL an especially effective language for context modelling.

Service-Oriented Context-Aware Middleware (SOCAM)[51, 52] have employed an OWL-based ontology that takes the idea presented within COMANTO one step further. SOCAM's ontology employs a similar division of a generalized upper ontology and a set of domain-specific low-level ontologies. The latter can be dynamically plugged and unplugged from the upper ontology, based on changes in the environment, such a user's movement from one domain to another. The upper ontology is broken down into four subcategories: person, location, computational entity, and activity. SOCAM's upper ontology is shown in Fig 3.1.

Chen and Finin proposes another ontology on context-aware applications in

their Context Broker Architecture (CoBrA)[53]. This ontology defines a set of vocabularies for describing people, places, agents and presentation events for supporting an intelligent meeting room system in a university building. With the use of OWL as the language for the ontology, concepts and relations could be clearly represented and shared between agents.

3.2.2 Context-aware System Frameworks

[54] defines several problems need to be addressed in designing standard data formats and protocols for context-aware infrastructure. The protocols used in the system should be universal applicable. While *Jini* is a negative example, *SOAP* is a good starting point for building a universal protocol. The context model should be comprehensive and rich enough to cover diverse sensors and assorted types of context. The fuzziness brought in by interpreting sensor data to context data is discussed. The concepts of precision, granularity, and accuracy are compared.

CASS (Context Awareness Sub-Structure)[55] by Patrick Fahy and Siobhan Clarke, is a context-aware server-based middleware used to support context-aware applications on small mobile devices. It avoids the problem of memory and processor constraints of mobile devices by making a stationary server as the context base, which gathers context information from a large number of low-level sensors and provides the information to mobile context-aware applications. Fig 3.2 demonstrates the CASS middleware architecture.

A major disadvantage of CASS is the apparent lack of an ontology for context

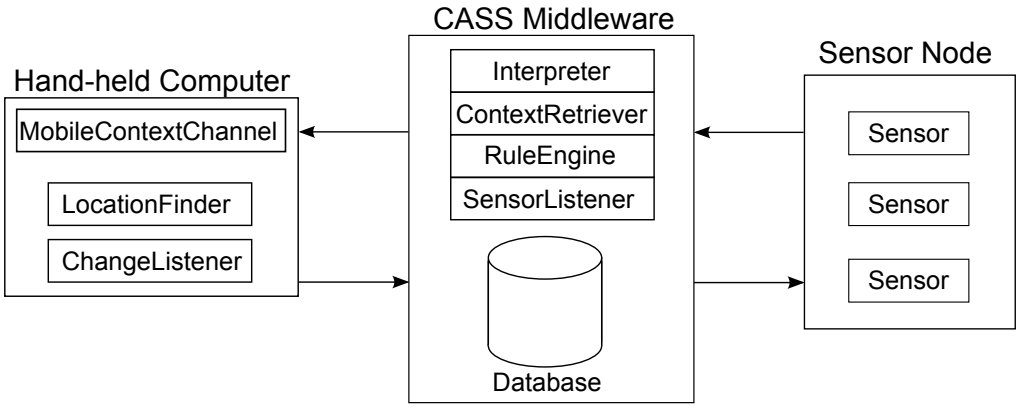

Figure 3.2: CASS System Structure

modelling. The database used for context storage is server-based, but the authors did not fully explain how the stored context should be modelled. It is not clear how applications on mobile devices are made aware of the table structures used within the SQL-based database, or how queries are made for context data within the database.

Most of the context-aware middleware and architectures presented thus far depend on a centralized approach to context awareness. However, the Hydrogen[19] architecture avoids this approach by introducing a distributed solution. The Hydrogen architecture shown in Fig 3.3 is divided into three layers: the Adaptor Layer, the Management Layer, and the Application Layer. The Adaptor layer is responsible for acquiring physical context information from the sensors and delivering it to the Management layer. Within the Management Layer, a context server stores all incoming contextual information about the current environment of the device, and shares this knowledge with other devices by using peer-to-peer communication.

However, the Hydrogen context-aware system architecture lacks two impor-

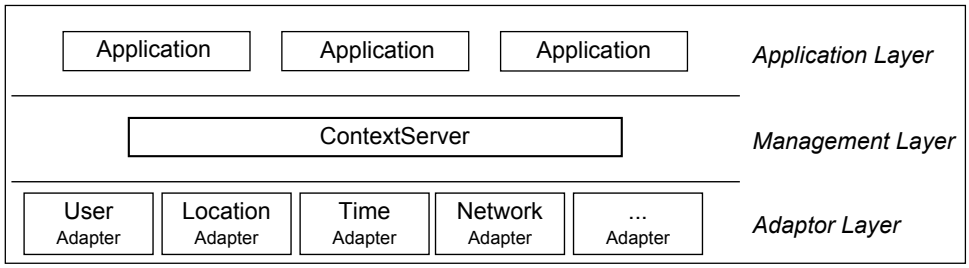

Figure 3.3: Hydrogen System Structure

tant components: an ontology on which context information is modelled, and a protocol by which context is shared between context servers located on different devices. Context within Hydrogen was limited to saving current time, the current location of the devices, the devices' identifier and type, the users' names, and information about available network connections. The authors had realized that the presence of ontology to model the vast amount of context available within a mobile environment is of high importance, and had set one of their future tasks to use CC/PP as the base for their context model.

Gaia[56] is a distributed middleware infrastructure which uses DAML+OIL as context model. These services permit applications to access and query Gaia's services and stored context such that user-centric and context-aware applications can be developed. Gaia thus acts as a coordinator between software entities and network devices, provides services related to location, context and events, and stores information related to the active space controlled by the Gaia kernel. Although Gaia's architecture was built for a context-aware ubiquitous environment, yet it is not suitable for an environment characterized by mobility. Queries for context information within the CFS required clients (applications) to be aware of the CFS's directory structure before clients can find the correct path to their

needed context. This mainly stems from the absence of an ontology on which to model the context information.

MundoCore[57] is a communication middleware specifically designed for the requirement of pervasive computing. To deal with the ultimate heterogeneity, MundoCore is built directly on operating system, uses its own communication protocols, and provides different language binding for services. The communication microkernel adopts a Advertise/Subscribe mechanism for intra-node communication. Pub/sub system is proved to be an efficient way to disseminate context information[58].

Solar[59] models context change as events, and devises a system of subscribing these events. Few applications want to work directly with raw data. It could be that the application only needs a portion of the data, the data is in wrong format, the data is inaccurate, or the data is incomplete and not useful without aggregating other sensor inputs. Thus, sensor data needs to go through several processing steps before it becomes the meaningful contextual knowledge desired by applications. Still, it is observed that many applications desire similar if not exactly the same contextual information. Therefore, Solar provides with a mechanism of operator graph.

In [60], the authors proposed an ontology-based generic context management (GCoM) model. The GCoM model facilitates context reasoning by providing structure for contexts, rules and their semantics. Context and context semantics in GCoM model are represented using the upper and the lower level ontology. Rules are represented using ontology compatible rule languages. Initial prototype

of the use of the model is created and the result obtained is promising.

[61] presents a uniform mobile terminal software framework that provides systematic methods for acquiring and processing context information from a user's surroundings and forwarding it to applications. To manage the context information systematically, the framework entities must have a common structure for representing information. An ontology is designed for this purpose[62]. To facilitate ontology sharing and communication, RDF is used as the description syntax. In this system, the inference rules are not set by programmers nor users, it uses supervised learning. A naïve Bayesian classifier is used to recognize all these high-level contexts from context atoms[63]. The classification results indicate that the naïve Bayesian classifier can extract situations fairly well, but also show that most results will likely to be valid only in a restricted scenario.

3.2.3 Context-aware Applications

Many interesting context-aware applications are developed, giving invaluable experiences to other developers. In [64], the authors evaluated the challenges and capabilities of combination of ontologies and rules in real-time ubiquitous applications. The project ec(h)o is a platform that provides audio guides in museums in an interactive way, overcoming the scheduling inflexibility of group tours by museum docents. Context-Aware Service Platform (CASP) [65] can aggregate and abstract context information. It uses ontologies to represent the information and rule-based reasoning to do validation and higher-level contexts derivation. Three distinct use cases are used to illustrate the usage of this system,

i.e., personalized content service, desk-sharing office environment, and person-oriented nurse-call management system. [58] addresses the pub/sub problem in context-aware computing. Users may subscribe to events published by services from 3 categories: location context, time context and event-preference context.

[66] presents an iterative development of a context-aware application including map showing, context-aware information provision, navigation and communication. An important application of context-aware devices is enhanced (augmented) reality.

[67] develops a context-aware tourist guide. The system uses a cell-based wireless communication infrastructure to collect location information. A number of use cases are studied in this work.

[68] introduces "discrete context-aware application", which means an application that deals with context information only in discrete domain. Then the author proposes a triggering mechanism for discrete context-aware applications. A naïve name-value pair way of representing context information is raised, and so is a matching rule. The matching rule is used to match predefined notes (or knowledge) and current context. If a match is found, the priori knowledge is triggered and shown to the user.

In [69], a system designed to derive Origin-Destination flows out of "network connection events" (mobile phone calls, SMSs, and Internet connection events). These events are simply extracted from anonymous dataset. The authors managed to extract the O-D flow information from the chaotic dataset after establishing several definitions and assumptions. When the outcome is compared to

a census ground truth, the high correlation indicates a good performance.

In [70], mobile persuasion systems are studied. This work explores the use of a mobile phone, when attached to an everyday object use by an everyday behaviour, becomes a tool to sense and influence that behaviour. By tracing the activity of drinking water, the system measures the daily intake of water amount and gives suggestions based on that. The drinking activity is monitored using a accelerometer with a threshold of 30 degree to capture one drinking motion. Then it uses a camera to detect the water level inside the bottle, de-noises the picture and measure the amount of water intake. The experiment also proposes a single-user game and a social game to facilitate the process of persuasion. Experiment results show people tend to have a healthier amount of water intake after the experiment, and the social game performs better than the single-user one.

OneBusAway[71] is a traveller-helper system deployed in Seattle. Only a web-based service is provided before an iPhone-enabled application is developed to facilitate location-based public transit information dissemination. This paper briefly explains the idea of real-time transit information system, and thoroughly studies the evaluation part of the system, gives a in-depth survey.

In [72], the authors presented an efficient offloading middleware, which provides run-time offloading services. The author also proposes an algorithm for partitioning instrumented classes into local classes and remote classes. This technique is very useful as mobile phones are usually the host of context-aware applications but the mobiles phones are still resource-restricted. Efficient of-

floading tasks to nearby devices can greatly promote the usage of context-aware systems.

Despite these context-aware applications, more and more researchers come to use a more systematic approach to ease the difficulty of building such context-aware applications. Guo[73] proposes OPEN—an ontology-based programming framework for rapid prototyping, sharing and personalization of context-aware applications. OPEN adopts three programming modes to accommodate different requirements from diversified users. They are Incremental Mode for skilled users, Composition Mode for middle-level users and Parameterization Mode for elementary users. Zhu [74] proposes a novel ontology-based context-aware framework where Application Context Models (ACMs) are instantiated at runtime by the engine. The engine then manages the whole life cycle of such ACMs to provide higher level context support for applications. In [75], the authors describe an ontology-based context model and a related context management system, providing a configurable and extensible service-oriented framework to ease the development of applications for monitoring and handling patient chronic conditions.

Ubiquitous context-awareness draws near as sensor-enabled mobile phones become more and more pervasively available. Campbell[76] raised the concept of people-centric sensing in this paper. People-centric sensing is defined in contrast to traditional mesh sensor networks. With the advent of sensor-enabled mobile phones, the recently thrived people-centric sensing becomes possible. The ubiquity of mobile phones solves the problem in traditional sensor networks. [77]

addresses the problem of high energy consumption when doing continuous sensing on mobile phones.

3.3 Ontology and Semantic Web Applications

D'Aquin's paper[78] is a decent summary on ontology and semantic web applications. It traces the history of the idea of semantic web, lists challenges and breaks down the challenges into concrete features and requirement for semantic web applications. A few types of semantic web applications are also investigated. [23] discusses many important features and solutions in the area of description logic. Some knowing in description logic (DL) can greatly help understand concepts in ontologies as DL is the underlying logical model for ontologies.

There have been many researchers working on the domain of distributed ontologies. But they mostly focus on the fusion of distributed ontology definitions, instead of breaking up a whole knowledge base into distributed ones where each contains a proportion of concrete information. Readers can refer to surveys[79, 80] for a review of this area of research. In the context of distributed ontologies, our work can be seen as a successive work that follows ontology fusion. After we successfully combined ontologies from distinct designers to form a super-ontology, the next step is to partition and relocate this monolithic knowledge base to distributed computing units for flexible uses.

To the best of authors' knowledge, there is only one group of researchers working specifically on the ontology extraction field. A series of work culminates at Bhatt's *MOVE*[81]. It is a distributed architecture for the extrac-

tion/optimization of a sub-ontology from a large-scale base ontology. The motivation of our work is more or less similar to Bhatt's, but after these many years from 2006 to 2012, with the idea and techniques of ontologies gradually coming to maturity, we are able to better understand the semantics of ontologies and its inference rules. In *MOVE*, the burden of determining extraction criteria is laid on sub-ontology developers. They require developers to "label" concepts and properties in the base ontology as "selected", "unselected" or "void". The architecture simply examines the consistency and completeness of the labeling without giving any guidelines or recommendations on how to achieve that. Moreover, their definition of the completeness of sub-ontology is restricted to 3 rules only, which however in the context of general query-answering procedure, is incomplete per se. The improvements in our work include:

1. We redefined the completeness of sub-ontologies in a more proper way.

2. Our system requires minimal effort for sub-ontology designers. Instead, the system enforces the completeness of sub-ontologies itself.

3. In addition to completeness enforcement, designers are given the option to trade part of the completeness for lightweightness of sub-ontologies.

3.4 Mobile Commerce and Recommendation Systems

Mobile commerce emerges around 2000[20]. It can be viewed as a subset of E-commerce and is usually defined as "any transaction with monetary value that is conducted via a mobile network"[21]. [22] surveys the features and char-

acteristics of current smart phones, putting an emphasis on the required and desirable features for mobile commerce. Though the smart phone market grows with unprecedented speed past the release of this paper, several points of this paper are proven to be quite insightful and predictive. Sumita [82] provides a mathematical model for comparing e-commerce via the traditional PC access with M-commerce which accommodates both the traditional and mobile access. However, there is no specific work on utilizing context-awareness in M-commerce settings.

As a starting point for our context-aware recommendation algorithm, the legacy recommendation algorithm for E-commerce should be investigated. Recommender systems are a powerful new technology for extracting additional value for a business from its customer databases[83]. These systems help customers find products they want to buy from a business. Recommender systems benefit customers by enabling them to find products they like. Conversely, they help the business by generating more revenue. Recommender systems are rapidly becoming a crucial tool in E-commerce on the Web. Existing recommendation algorithms can be basically classified in two categories: content-based methods and collaborative filtering (CF) methods.

Content-based recommendation method has its root in data mining, information retrieval and information filtering. The essence of the method is to find association rules in databases. [84, 85] are considered to be the biggest contribution in this field. The authors considered the problem of discovering association rules between items in a large database of sales transactions. They seek to dis-

cover co-purchased products and return top-N recommendations. The Apriori and AprioriTid algorithms proposed in this work are proved to be able to discover all significant association rules in large database.

The developers of one of the first recommender systems, Tapestry[86], coined the phrase "collaborative filtering". The fundamental assumption of CF is that if users X and Y rate n items similarly, or have similar behaviors (e.g., buying, watching, listening), and hence will rate or act on other items similarly. Collaborative filtering, being the most successful recommender technology to date, recommends products that are similar to the ones that the target customer has purchased. This similarity can be calculated in many ways, and thus classifies the family of CF algorithms into user-based CF and item-based CF.

User-based CF has its representative work as GroupLens[87], Video Recommender[88], and Ringo[89]. In user-based CF algorithms, the similarities between users are calculated as a metric of the rating matrix. The recommendations are given based on similar user's preferences. Item-based CF[90], however, measures the similarities between items first. For each user, predicted rating values are given on all the items, and the item with highest (top-N) predicted values are returned. Some of the new implementations of these recommendations include[91, 92]. It is believed that item-based CF algorithms generally provide better scalability and higher accuracy in giving recommendations. Our algorithm is rooted in item-based collaborative filtering.

[93] employs the technology of Collaborative Filtering from e-commerce to context-aware systems. By observing user's past choices made, this system tries

to predict user's preferences. Each preference is associated with a specific context, and previous choices made in the same context as to current context contributes more to the prediction of user preference. The paper discusses two issues here: 1. how to manage context in the user profile in terms of data modelling and storage, and 2. how to measure the similarities between contexts.

3.5 Chapter Summary

In this chapter, different research aspects of this thesis are considered. Extensive literature reviews are given for these aspects and the strength and weaknesses of different approaches are analyzed. Specifically, current mobile application platforms are surveyed. This survey calls for a more sophisticated platform with the support of context information management. After comparing various context modelling techniques and context managing frameworks, the ontology model and the reasoning system is chosen to perform context manipulation. The domain of ontology partitioning/extraction is covered and the shortcomings of existing methods are discussed. This puts highlights on our work on partitioning the ontology database while maintaining the completeness. Lastly, the context-aware recommender systems are reviewed and we explained why a new context-aware recommendation algorithm is required and how we can benefit from this new algorithm.

Chapter 4

An Ontology-based Test Bench for Context-awareness

4.1 Introduction

Experimental methodology in the area of pervasive computing research has long been an unsolved problem[94]. There are no available benchmarks or conventional experiment design process. The common approach of doing research is to build a real system and test out ideas in real world. However, this approach significantly prolongs the development phase. In order to test out a single small idea in this area will require a whole system being built. Moreover, applications built for one purpose usually cannot be re-used for another purpose.

There are some benchmarks in the field of ontology reasoning, such as LUBM[95], UOBM[96], and BSBM[97]. However, they do not cover the domain of pervasive computing, so experiments done on those benchmarks are not representative

enough to guarantee a promising result in real world. LUBM[95] features an ontology for the university domain. It can be populated with synthetic OWL data scalable to an arbitrary size. 14 extensional queries, a variety of properties, and several performance metrics are introduced as well. The LUBM is usually used to evaluate systems with different reasoning capability and storage mechanisms. In the domain of a university, the most important concepts are schools, faculty members, students, and the modules they take. This is a very static hierarchical structure. The interconnection of concepts are restricted to the *TakeModule* property. University Ontology Benchmark (UOBM)[96] from IBM extends LUBM in several perspectives. It compares RDF storage along with reasoning capabilities of ontology management systems. OWL DL and OWL Lite are supported in UOBM. The Berlin SPARQL Benchmark (BSBM)[97] compares the performances of storage systems that expose SPARQL endpoints. Sadly, none of these benchmarks has extended the domain into the area of context-awareness or mobile applications.

In a typical context-aware application, the concept and relationship structure is quite different from that of a university. The concepts cover a much wider area, including locations, people, activities, mobile phones, cars, computers, foods, bank cards, and many more entities that are linked to our daily lives. The hierarchy is also significantly different. The class hierarchy tree of *Location* is very deep because locations can be defined using different granularities. When specifying the location of a person, we can use the granularity of latitude/longitude, room, floor, building, district, city, or even country. Other class

hierarchies are usually relatively shallow. The class of *Person* can be classified into *Man* and *Woman*. In another perspective, it can be classified into different age groups. But these hierarchies are not further classifiable. The relationships between concepts in the domain of context-awareness are also much more complicated than on the domain of university.

To sum up, we are in urgent need of a test bench that describes the domain of mobile context-aware applications, so that experiments done on the test bench can be generalized into other context-aware applications.

A complete ontology-based test bench should incorporate many components. Firstly, it should include an ontology that captures the domain of discourse. Specifically, it should include the concepts and relationships in the domain of context-aware applications. For a basic form of the test bench, only an upper ontology that describes the highest level is enough. But for a more thorough testing, one can extend the upper ontology to include application-specific ontology to better mimic the reality. Secondly, the test bench should be able to generate randomized concrete data. To make the test bench scalable, the data generator should be able to take as input the desirable size of the database, and then generate the concrete data. The generated data should not be arbitrarily generated. For example, it is absurd to generate a person who has 100 children. It is also incorrect to generate a database where everyone's current location is unique, even though this is a much easier way of generating people's location. To achieve these common senses, a set of rules are to be followed in the geneartion process. In addition to this, a complete test bench should consider the possible

query types the server would receive. The responding time of a simple query and the time of a complex query can be quite different. By envisioning the possible query types, we can get more reliable estimates of responding time.

In the following sections, we describe how we have constructed the test bench.

4.2 Ontology Construction

4.2.1 Mobile Applications Survey

The most important component of a test bed is the ontology that models the domain of context-aware systems. The test bench ontology should reflect the situation in real-life applications. The typical concepts and relationships in context-aware applications should be correctly reproduced in the upper ontology. This upper ontology can then be further extended with application-specific sub-ontologies to account for different application requirements.

Since current mobile applications are akin to context-aware applications (To be precise, some of them are already context-aware though not to a high extent), this work intends to build a test bench for context-aware systems by extracting features from existing mobile applications.

In our work, 221 popular (with at least 10,000 installations) Android applications are surveyed to model the knowledge domain of mobile applications, among which 176 have the design that embraces context information sharing. These 221 apps are selected in the following procedure: First we visit Google Play (the application market) under different categories with applications sorted

by their popularity. The "Games" category is not considered because they are mostly not context-aware and they are difficult to analyze. Then we select all applicatioins with more than 10,000 installations while excluding some obvious duplications. For example, one application may come with a paid ad-free version and a free version with advertisements. Only the paid version is considered. Some mobile banking applications are also very similar to each other with the only difference in the bank name. In this case, only one application is randomly selected in our survey. This test bench is by no way exhaustive, but attempts to find some patterns in current mobile applications, and to apply these patterns to future context-aware systems. Applications are chosen from 20 different categories that include *Business, Communication, Finance, Health, Media, News, Social*, etc. 101 of them are paid applications while the other 120 are free. Only English-language applications are surveyed.

The survey is carried out for each application in the following procedure.

1. The first step is to extract the use case scenarios of the application under concern. This may be deduced from the description on Google Play. When the description is too vague as to the functionalities, the application is downloaded and tested to model its use cases.

2. Abstract the use cases into SPARQL queries, with the necessary classes and properties defined at the same time. If an application has too many functionalities, only the few that is most relevant to context-aware computing and the major usage of the application is extracted. For example, for a use case that is to answer users how much calories is contained in 100

grams of the asked food. This use case is then abstracted as `SELECT ?x WHERE { thefood ns:hasCalories ?x. thefood rdf:type ns:Food }`, where `thefood` is the user input choice of recipe. The relevant classes and properties are constructed at the same time, including the class `ns:Food` and the property `ns:hasCalories`. After the first round, we have generated many classes, properties, as well as query types. Admittedly, the enumeration of queries is neither exhaustive nor precise. But in large it captures the most important behaviours of these applications.

3. The next step is to integrate the classes and properties into a whole ontology with hierarchies, restrictions, and characters. This step is done as a collaborative work. Multiple versions of the ontology are proposed by the authors and then revised to get the final one, aiming to represent an expert's view.

4. The last step is to examine the appropriateness of using this ontology in those applications. In the mean time, the dependencies among classes are also examined. Together proposed is a set of rules that represent common senses in the quantity restrictions. These are helpful in the synthetic concrete information population, which will be detailed later.

Table 7.1 in Appendix shows part of the survey results. Two to four applications for each category are shown in this figure based on their suitability of converting to context-aware applications.

4.2.2 Ontology on the Domain of Context-awareness

Ontology modelling provides applications with a mechanism to share context information, and a common language to talk to each other. By joining the vocabularies of all applications, we form the knowledge domain of our test bench shown in Fig 4.1. The definition of 250 *owl:Class*, 147 *owl:ObjectProperty* and 94 *owl:DatatypeProperty* are presented in this domain ontology. The development of this ontology is aided by Protégé[1]. As shown in Fig 4.1, *ContextEntity* is the ancestor for all classes, which is presented at the top of the figure. Its direct children (sub-classes) include *Location, Activity, Person* and *OtherEntity*. Class inheritance relationships are shown with hollow arrows while properties are shown with solid arrows pointing from the domain class to the range class. Due to the limited space, only a very small fraction of the ontology is shown here. One can contact the authors requesting the full OWL file to get a complete view of the ontology.

The most important context information within context-aware systems is location. In fact, most early context-aware systems are essentially location-aware applications. This is partially because the ubiquity of GPS (Global Positioning System) sensors in current smart phones. Though GPS sensor readings directly extracted from the sensors are not immediately useful, our ontology still models this reading as an object of class *GPSLocation*, with *DatatypeProperties* of *LatitudeOf* and *LongitudeOf* storing the reading value. Currently, the typed locations in our test bench include *IndoorLocations* and *OutdoorLoca-*

[1]http://protege.stanford.edu/

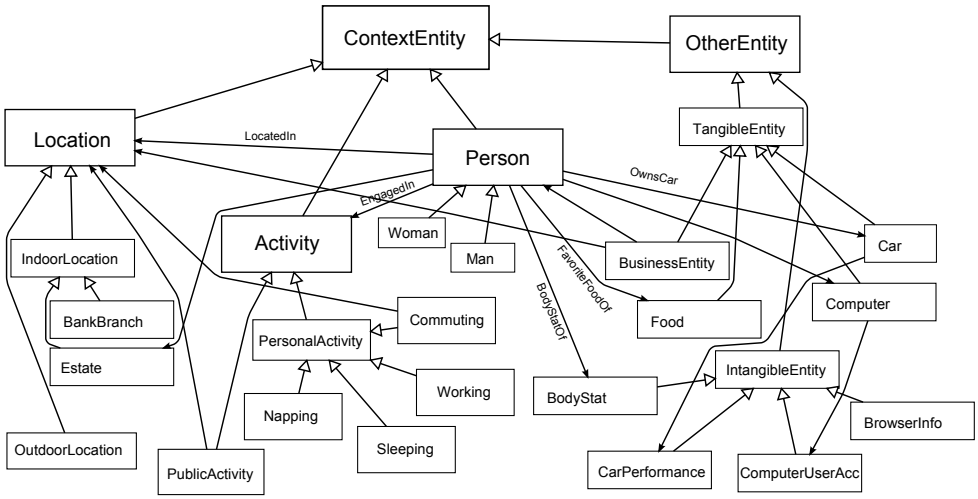

Figure 4.1: (partial) Knowledge Domain of Test Bench

tions. *IndoorLocations* can be further classified into bank branches, cafe, estate (including houses and apartments), home, office buildings, shopping malls, and subway stations. *OutdoorLocations* include bus stops, landmarks, high ways, lanes, pathways, and roads.

The knowledge of a person's *Location* can be very helpful in deducing other contexts. For example, the individuals of class *Location* may have *LocatedIn* relationships. Since the *LocatedIn* is a transitive property, a person located in a location of finer granularity can be deduced to be also in the location of more coarse granularity. Here is another example. A person is currently seen in some building in Singapore. After several hours, if there are not departure flights from Singapore airport, it can be safely deduced she is still in the same city. The knowledge of a person's *Location* can also be used to deduce the person's current activity. When a professor and several students under the supervision of that professor are presented at a same meeting room, it can be deduced that

they are currently having a meeting.

In the ontology developed, additional classes are designed to simulate the effect of distributed composition of ontology. We envision our system as a platform for user sharing and contribution where all the data and data structure are not determined by one authority. So a centralized server with all vocabulary defined in central control is not reasonable in real deployment. The reasonable assumption is: Several developers each independently designs the ontology for the same domain. When the developers made their application available to others, they need to conform to a ontology fusion rule to make their individually designed ontologies compatible with others. In our work, we do not discuss further on the ontology fusion rules, but the basic method of ontology fusion is done with the help of *owl:equivalentClass* and *owl:sameAs*.

In order to account for the difference in the generated ontology, we need to deliberately add some elements to the database so that the outcome more resembles a real database with distributed composition. Here we add triples using OWL primitive *owl:equivalentClass* for the composition of classes, and *owl:sameAs* for the composition of individuals. Classes that are not tightly coupled with a single person, and do not have authoritative UID (Unique Identifier) to eliminate duplication, are subject to the modification. Individuals in these classes are randomly linked with *owl:sameAs* primitive so that statistically half of them are linked to other individuals.

4.3 Other Components of the Test Bench

4.3.1 Synthetic Concrete Information

In order to carry out query experiments, concrete information expressed in the ontology definition must be generated. Similar to the construction of the ontology, the concrete data should also be created in a way that is most similar to the real world cases. Synthetic OWL database is scalable to arbitrary size so as to simulate arbitrarily large scenarios. In LUBM, this scalability is achieved by varying the number of universities and the number of departments in each university. In our test bench, the size of the synthetic data is determined on 3 variables— the number of *Locations*, the number of *Person*, and the number of individuals from other independent classes.

Data generation is carried out by a Data Generator we have developed. Developing this software is surprisingly not so easy. On one hand, the generated data should be essentially random; on the other hand, each piece of data should be connected to other concepts even before the other concepts are generated. The relationships between concepts reduces the degree of freedom, and thus introduces more complexity. To address this problem, we analyze the dependencies among classes. We aim to make the data as realistic as possible so a set of rules that represent common senses are abided by in the generation process.

In the generation process, all *individuals* are classified into several groups. They are, independent non-scalable individuals, individuals of class *Person*, individuals of class *Location*, other independent individuals, Person-dependent indi-

viduals and non-Person-dependent individuals. When the input to the generator program is given, independent non-scalable individuals are generated in the first step. In our test bench, this includes *Genre*, *PaypalPaymentMethod*, *VisaPaymentMethod*, *MastercardPaymentMethod*, *PhonePlan*, *PrivacyPolicy*, *Privilege*, *ChineseZodiac*, and *WesternZodiac*. In fact, these independent non-scalable individuals are similar to the concept of *Classes* because they are actually determined prior to the size of concrete data. We can opt to re-define these individuals as classes in the ontology. After weighing the different choices, these individuals are determined to be more of a flavour of concrete data, so they are generated as individuals. The second step is to generate individuals of class *Person* and *Location* and other independent individuals. Note that the three variables that determine the size of the concrete data are the number of individuals of class *Person*, *Location*, and other independent individuals. Following this, for each *Person* generated, all the person-dependent classes are traversed to generate individuals. The relationships between the newly generated individuals and the person under consideration are added. This process is guided by the rules that reflect the real world. As mentioned in the first section in this chapter, we use a quantitative rule to restrict the number of children of a parent. With the generation of each person-dependent individual, another rule set is checked to determine if any non-person-dependent individuals should be generated.

A total of 180 *Classes* are subject to the grouping. This number is smaller than the total number of *Classes* because other *Classes* are of more generalized concepts. Only the *Classes* that have the most specific meaning, or rather

the "*leaf Classes*", are subject to concrete data generation. That is, class *Person* are not grouped, but classes *Man* and *Woman* are groupped. These classes are grouped into 9 *independent non-scalable Classes*, 2 *Person*, 14 *Location*, 45 *other independent Classes*, 63 *Person-dependent Classes*, 47*non-Person-dependent Classes* (among which 36 are indirectly dependent on *Person* and the rest are dependent on other *Classes*). The data are generated in *Turtle* language as described earlier in Section 2.3.1. In the generated data set, we will have the following independent non-scalable individuals: 5 *Genre*, 20 *PhonePlan*, 12 *ChineseZodiac*, 12 *WesternZodiac*, etc. They are linked to *Person* individuals later. After determining the number of *Person* individuals, for each *Person* generated: The *Person* has 1 *GPSLocation*, 1 *Direction*, 1 *Contact* (which has 2-4 *SNSAccount*, 0-2 *ReaderAccount*, 1-3 *EmailAccount*, 0-1 *ForumAccount*, 0-7 *IMAccount*, 0-2 *OnlineDocAccount*, etc.), 0-1 *BibleReadingPlan*, 0-1 *WorkoutPlan*, 0-1 *BrowserHistory*, 1 *CallLog*, 0-8 *FinancialEntities*, and many more. These person-dependent individuals can have many non-person-dependent individuals further.

Finally, the generator generates a 342-people knowledge base including a total of 33841 *individuals*, 53607 *ObjectProperty* assertions, and 23312 *DatatypeProperty* assertions.

4.3.2 Testing Queries

The performance of a context-aware system boils down to the query responding speed. Because the connection speed between client and server is fixed regardless

of the way server is organized, the performance of such a client-server system is solely measured by the query processing speed on the server side. Query processing speed of a system is highly dependent on the query pattern. This is especially true for ontology-based servers. Two query sequences can have very different query processing performance. The difference can be as high as orders of magnitude in time. So the testing queries are another important component of the test bench.

Recall that in the second step of the mobile application survey, the use cases of the mobile applications are abstracted into SPARQL queries. So from that step, we have already prepared the queries from all the applications we have surveyed. Some similar queries from the same category of application are combined together and some queries that are less important to the category of application are removed. The queries are also modified so that a more generic form is reserved in the final testing queries of the test bench. Since the queries are abstracted from real mobile applications, the performance measured using these testing queries are more reliable than the ones measured using other benchmarks.

Though we have surveyed 20 categories of applications, a total of 9 categories are considered here to produce the testing queries. Specifically, these 9 categories are *Business, Communication, Health & Fitness, Lifestyle, Media & Video, Medical, News & Magazines, Shopping*, and *Transportation*. For each category, we have prepared at least 6 query sequences. These query sequences are selected such that both simple and complex query structures are included. Table 4.1 exemplifies one query sequence for each category. The query sequences

are given in SPARQL language while resources are represented using the Turtle language for concise presentation.

Table 4.1: Selected Testing Queries

Business

```
PREFIX ns:  <http://dsa.nus.edu.sg/cenzhe/ontologies/ContextEntity#>
SELECT DISTINCT ?x ?y WHERE { ns:Man0 ns:ContactInfoOf _:a .
    _:a ns:EmailAccountOf _:b .
    _:b ns:ContactsOfEmailAccount _:c .
    _:c ns:ThePersonOfContact ?x .
    ?x ns:ScheduleOf ?y .
    ?y a ns:Calendar }
```

Communication

```
PREFIX ns:  <http://dsa.nus.edu.sg/cenzhe/ontologies/ContextEntity#>
SELECT DISTINCT ?x WHERE { ns:Man0 ns:ContactInfoOf _:a .
    _:a ns:IMAccountOf ?x }
```

Health & Fitness

```
PREFIX ns:  <http://dsa.nus.edu.sg/cenzhe/ontologies/ContextEntity#>
SELECT DISTINCT ?x WHERE { ?y ns:DinnerOfDiet ?x .
    ?y a ns:Diet .
    ?x ns:CalorieOf ?z .
```

Continued on next page

Table 4.1 – continued from previous page

`FILTER ( ?z > 0 ) FILTER ( ?z < 50 ) }`

Lifestyle
`PREFIX ns: <http://dsa.nus.edu.sg/cenzhe/ontologies/ContextEntity#>`
`SELECT DISTINCT ?x WHERE { ?x ns:RentalValue ?y .`
`?x a ns:Apartment .`
`FILTER ( ?y >= 500 ) FILTER ( ?y < 550 ) }`

Media & Video
`PREFIX ns: <http://dsa.nus.edu.sg/cenzhe/ontologies/ContextEntity#>`
`SELECT DISTINCT ?x WHERE { ns:Man0 ns:ContactInfoOf _:a .`
`_:a ns:ITunesAccountOf ?x }`

Medical
`PREFIX ns: <http://dsa.nus.edu.sg/cenzhe/ontologies/ContextEntity#>`
`SELECT DISTINCT ?x WHERE { ns:Man0 ns:HaveDisease _:a .`
`_:a ns:PrescriptionOf _:b .`
`_:b ns:Dosage ?x }`

News & Magazine
`PREFIX ns: <http://dsa.nus.edu.sg/cenzhe/ontologies/ContextEntity#>`
`SELECT DISTINCT ?x WHERE { ns:Man0 ns:ContactInfoOf _:a .`
`_:a ns:ReaderAccountOf ?x .`
`?x a ns:GoogleReaderAccount }`

Continued on next page

Table 4.1 – continued from previous page

Shopping

```
PREFIX ns:  <http://dsa.nus.edu.sg/cenzhe/ontologies/ContextEntity#>
PREFIX xsd:  <http://www.w3.org/2001/XMLSchema#>
SELECT DISTINCT ?x WHERE { ns:Man0 ns:HaveGroceryStuff ?x .
    ?x a ns:Coupon .
    ?x ns:AmountOfDiscount ?y .
    FILTER ( ?y > 0.7 ) }
```

Transportation

```
PREFIX ns:  <http://dsa.nus.edu.sg/cenzhe/ontologies/ContextEntity#>
SELECT DISTINCT ?x ?y WHERE { ns:Man0 ns:OwnsCar _:a .
    _:a a ns:Car .
    _:a ns:StartParkingTime ?x .
    _:a ns:LocatedIn ?y }
```

4.4 Evaluation

The test bench is constructed solely for the purpose of testing in the domain of context-aware systems. Therefore, it is not meant to be an overall Semantic Web benchmark. It is limited to the particular domain represented by the ontology it uses. Be reminded that the motivation of building such test bench is to mimic the data structure of real context-aware systems. The insights we gained through

Table 4.2: Structural Differences between our test bench and LUBM

	Our Test Bench	LUBM
# of *Class*	250	43
# of *ObjectProperty*	145	25
# of *DatatypeProperty*	94	7
# of *Individual*	33841	17174
# of *SubClassOf* Axioms	238	36
# of *EquivalentClass* Axioms	13	6
# of *DisjointClass* Axioms	2	0
# of *General Concept Inclusion*	13	2
# of *SubObjectPropertyOf* Axioms	138	5

building such test bench may be even more valuable than the test bench itself.

Our test bench is for the domain of context-awareness and existing benchmarks (e.g. LUBM) is on the domain of a university. Qualitatively, LUBM is a *descriptive* benchmark. By defining a hierarchy of classes in the domain of a university, LUBM describes departments, research staff, students, modules and many other elements of interest. On the contrary, our test bench is a *functional* framework. From the very beginning of designing this test bench, it has kept applications and queries in mind. The objective is not to describe concepts and facts in this domain, but to answer specific types of queries. If certain concept is not used by any queries, it is not modelled in our test bench. This qualitative difference reflects the different usage of the two test benches.

In the following texts, we quantitatively evaluate our test bench.

We first measure the ontological complexity of the test benches. From Table 4.2 we can clearly see our test bench is more sophisticated than LUBM. In all measures, our test bench has a larger number. Except for the number of *Individuals* which reflects the size of ABox/concrete database, all other measures

reflect the complexity of TBox/vocabulary. The differences in these numbers manifested that the domain of context-awareness is significantly different than the domain of a university. Specifically, the increase in the complexity of vocabulary contributed to this difference. This difference is most conspicuous for *Properties*-related axioms. This is because a lot of data relevant to a user or a device appear as key-value pairs. In ontologies, key-value pairs are represented as Datatype Properties. The connection between different objects is represented as Object Properties, and as such the number of ObjectProperty is also much larger than that in LUBM.

Another metric we measure is the depth of class taxonomy. Suppose the root class is of depth 1, its direct subclass is of depth 2, we can calculate the depth of all classes. An ontology with a smaller average depth is considered to be *flatter* than another ontology with deeper class taxonomy. This is a frequently considered parameter when designing an ontology, and this feature can significantly impact the query-answering performance. Fig 4.2 shows the cdf (cumulative distribution function) of the depth distribution. The average depth of classes in our test bench is 5.59 while in LUBM is 3.62. With an ontology that is averagely 2 layers deeper, we can expect these two test benches will give very different performances when answering queries. This difference again necessitates the introduction of a new tool—our test bench—in building context-aware systems.

Finally we also measure the loading time of our test bench and LUBM. As presented in Fig 4.3, we plot the loading time versus the number of triples

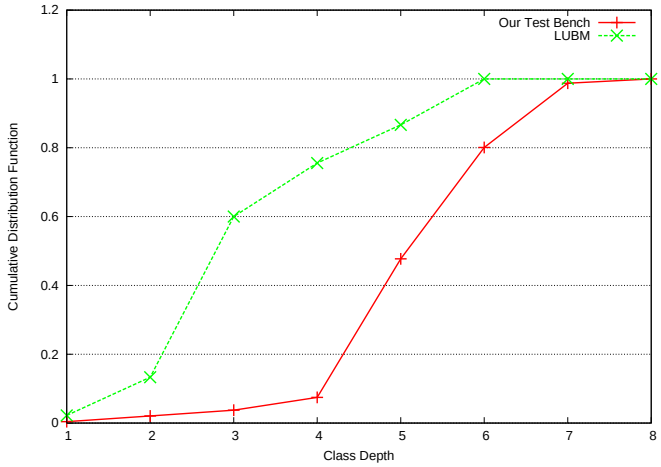

Figure 4.2: Class Depth Distribution

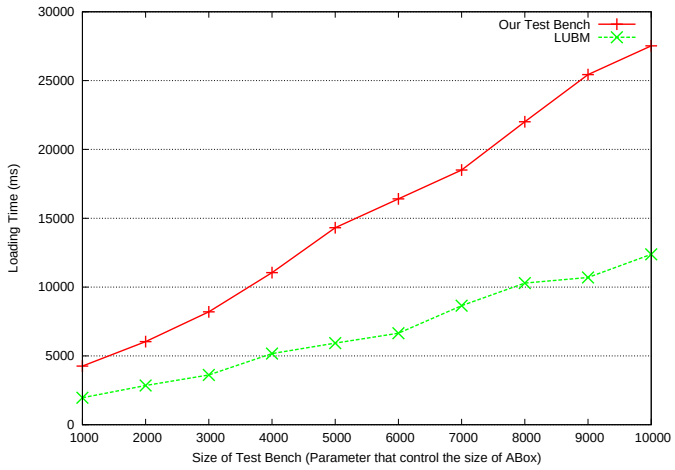

Figure 4.3: Ontology Loading Time

in the test bench. Results show that the loading time is linear to the size of the ontology, but the curve of our test bench, which is of higher complexity, is steeper. The difference in the slope of the curves from another perspective justifies the necessity of building our own test bench. The two different domains have different features, and these features cannot be masked by merely adjusting the size of the test bench.

Regarding to the query-answering performance, it is postponed in Chapter 5

to discuss. In Section 5.5, we will be measuring not only the query-answering performance of our test bench and LUBM, but also the performance of our newly introduced partitioning algorithm.

4.5 Chapter Summary

In this chapter, we first demonstrated motivation of building an ontology-based test bench on the domain of context-aware applications. Then we constructed a novel test bench covering the domain of context-aware applications. To the best of authors' knowledge, there have been no ontologies that are specifically built for the domain of context-awareness. Thus, evaluating the algorithm on other test benches may be biased if the system is data-dependent. Through an extensive survey of state-of-the-art mobile applications, the domain of knowledge in this area is modelled as an ontology. A large amount of synthetic concrete information composed using this vocabulary is populated, together with several sample applications, query sequences and knowledge duplicates.

Compared with LUBM and UOBM, our test bench has the following contributions:

1. The test bench is constructed with real mobile applications. Other ontology-based benchmarks usually build the ontology merely from common senses and experiences. After all, our test bench is built for the purpose of being as realistic as possible, while other benchmarks emphasizes on utilizing OWL language structures and SPARQL query types.

2. Our test bench considered the feature of distributed composition of ontologies while other benchmarks assume centralized ontology composition. This allows our test bench to evolve as the domain of mobile applications change. When the domain is enlarged or shifted, updates to the ontology can be made by application developers in addition to original designer of the test bench (which is us!).

3. Our work clearly formulates the concrete data generation process. Our method distinguished different types of classes and individuals, thus giving rise to a dependency-based generation. This experience can be useful for any other concrete data generation process.

Chapter 5

A Distributed Computing Scheme for Better Scalability

5.1 Introduction

This section addresses the scalability issue of ontology-based context-aware systems by breaking the monolithism of knowledge bases, and thereby making the divide-and-conquer approach applicable.

The final quest of our research is to enhance feasibility of ontology-based context-aware systems, specifically in the sense of bridging the gap between real-time application requirements and the slow reasoning speed of ontology servers. Approaches can be either to improve the processing speed of a single server, or to provide a viable means to distribute tasks among many servers. Regarding the first approach, over decades researchers have been working to build an efficient ontology reasoner. Implementing the tableaux algorithm described in [23], many

state-of-the-art ontology reasoners are built, such as *RacerPro, FaCT++, Pellet* and *HermiT*. Though we are pleased to see the advances in reasoning speed, we have to admit that a single reasoner still cannot fulfil high-load real-time tasks[24].

This motivates the development of second approach: task distribution. To better illustrate the goals and challenges of this approach, an example is introduced. Suppose we have developed a digital butler system that stores all personal information of users and gives all kinds of assistance to them. The whole knowledge base is huge as it has to store every aspects of users as well as other knowledge that are independent of users. "other" knowledge include the locations the user can be, the activities that the user can engage, and other common senses. When the system scales up, a single-server solution is not applicable. The naïve solution is to clone servers, by copying both the front-end and the database behind. Server cloning requires copies of the whole ontology, which can be very cumbersome. Moreover, it requires a large amount of real-time synchronization between all these copies because updates to one server should be pushed to all other servers. In our example, the knowledge base of all aspects of the users should be cloned, and any trivial updates like the location change of certain user should be forwarded to mirror servers. Otherwise, some users may get wrong query response from servers that are incomplete or obsolete. These two problems can be alleviated if we can further classify tasks (specifically, queries in our work), and allocate each category of tasks to a specific server extracted for this specific purpose. Queries from a same context-aware application naturally

form a category of queries. In the following texts, we use the word "application" interchangeably with a category of tasks or queries. In our example, this solution is applied by placing sports-related data in one server, finance-related data in another, so on and so forth. Therefore, the location change of users may be irrelevant to the server hosting finance-related applications, and thus we can reduce the amount of synchronization between mirror servers.

But how to define "sports-related" and "finance-related"? It is much more complicated to define this in ontology databases than in relational databases. In relational database systems, we can simply construct the database for the extra server by extracting a few tables from the original database or a bunch of tuples that matches a specific query. These can all be done in a neat and elegant way. However, ontology databases are constituted of many interconnected statements which entail special treatment. A statement is a triple formed of a *subject*, a *predicate* and an *object*. While many statements are readily available in the database, additional statements can be derived following the semantics of the ontology. In order to construct a sub-ontology to handle a specific category of queries, we are facing many challenges and these are summarized and termed as the monolithism of ontology databases:

1. It is hard to determine the necessary statements that should be extracted for a functionality. One seemingly irrelevant statement may be useful for deducing relevant information.

2. Generally, only statements that are raw and NOT deduced should be in-

cluded in the extraction as such deduction can sometimes increase the size of the database exponentially.

3. After extraction, how to keep the extracted database synchronized to the central one is also a challenge.

We tackle these three problems by:

1. We proposed a completeness-proved algorithm to determine the necessary triples that fulfil certain functionalities.

2. We migrate only existing triples (i.e., not including inferred triples) in order to restrict sub-databases within controllable size.

3. A fast algorithm is employed to coordinate synchronization traffic.

Using the algorithm described in this section, we can extract application-specific sub-databases from the whole knowledge base. It is also proved that the extracted sub-database can perfectly accommodate queries from that specific application, which is also known as the completeness of the sub-database (or algorithm). The algorithm also covers the synchronization process. When an update of information is received at one server, it can be quickly decided (without going through an ontology reasoning process) whether this update should be delivered to other servers.

Despite our algorithm, one might come up with a more straightforward way of building such a system. That is, one could have a system in which the entire knowledge base was replicated, and accept that queries performed in different

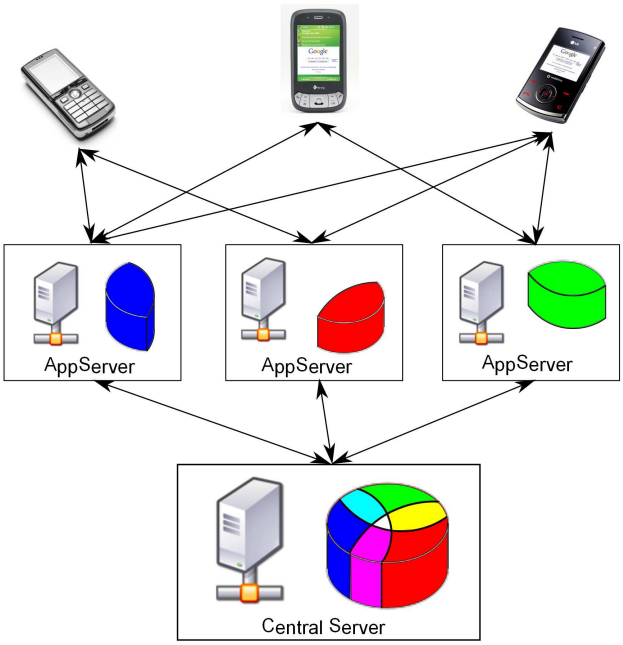

Figure 5.1: System Structure

places might generate different results for data that is in the process of being updated. However, this approach suffers from following problems: 1. This solution requires significantly more resources because the whole knowledge base is to be replicated. One central supercomputer is acceptable but an array of supercomputers is frustrating. 2. With a bigger ontology, the query processing speed would be slowed. Therefore, we will have a longer delay in getting responses from this solution. 3. We will have a lot of unnecessary updates. Even if we accept this redundancy, servers will consume time and power to handle those unnecessary updates. To sum up, this easy-to-implement approach is not a satisfactory solution.

Fig. 5.1 illustrates the topology of our system where *AppServers* hold sub-databases for a single application and *Central Server* holds the whole database.

However, it is observed that the extracted sub-ontologies are usually larger than enough—many triples are seldom used for inference but must be present in order to ensure absolute completeness. Sometimes these triples take up a large proportion of the extracted sub-ontology, incurring higher storage and computational costs. We then introduced a tunable parameter. By adjusting this parameter, we are able to adjust the "degree" of completeness. When this parameter is set to zero, the system retrogresses to a naïve implementation of only labeling and filtering. The produced sub-database is minimal as it contains only the triples explicitly mandated by application developers. When this parameter is set to infinity, the filter expansion algorithm then can guarantee semantic completeness of sub-databases. However, the size of sub-database is considerably larger than the minimal one. After setting the parameter to a intermediate value, the algorithm is no more complete but we established a checkpoint method to compensate for it. Every once in a while, an ontology reasoning is performed and sub-databases are updated to achieve temporary completeness.

The test bench introduced in Chapter 4 is then used to test the performance of our algorithm. To further justify the validity and performance of our algorithm, a typical ontology benchmark *LUBM*[95], though not specifically designed for pervasive computing area, is employed in order to generate comparable results. Results show that our algorithm can achieve a drastic improvement in terms of processing speed of sub-database extraction and synchronization. Compared to the speed boost in thousandfold or even more, the cost in ontology storage is acceptable. It is also shown that, without losing any quality in the

answers to queries, a much smaller cost can be achieved by carefully tuning the trade-off parameter.

This chapter is organized as such: Section 5.2 explains all 3 phases of the extraction algorithm from the starting point of the definition of domain of discourse. A recommendation on how to decide the domain of discourse for an application is also included. Section 5.3 proves the completeness of the algorithm. Section 5.4 introduces the trade-off between completeness and lightweightness of sub-databases. We evaluate our algorithm in Section 5.5 and conclude in Section 5.6.

5.2 Algorithm of Extraction and Synchronization

This section explains the algorithm to extract information from the whole knowledge base in order to answer a category of queries. It also covers the update procedure after the initial extraction. The feature of this algorithm is that we use a set of filters to determine if a triple is relevant to the category of query.

This algorithm is designed for the dialect of OWL DL. We describe the algorithm in 3 phases—PREPARATION phase, SETUP phase, and UPDATE phase. In PREPARATION phase, a set of filters is derived from the domain of discourse. In SETUP phase and UPDATE phase, these filters are applied to all existing triples and newly-updated triples. This process is illustrated by Fig 5.2.

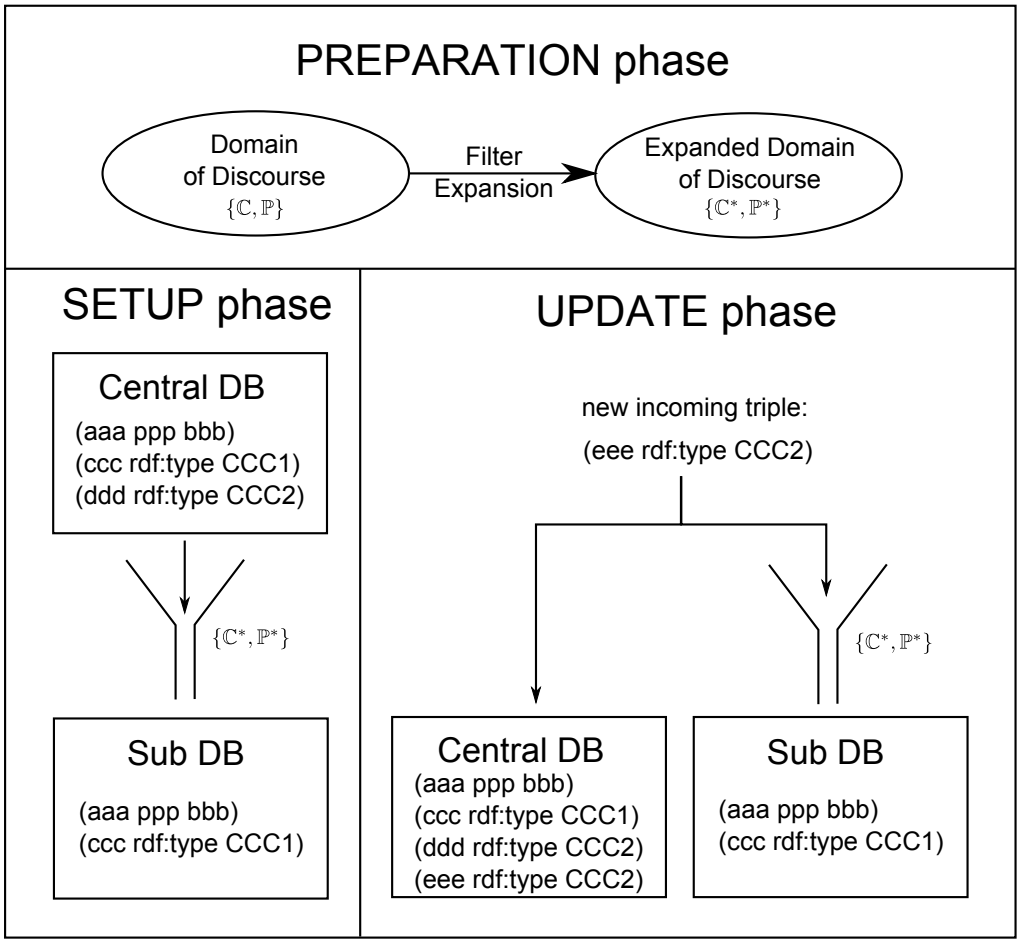

Figure 5.2: Illustration of the Partitioning Algorithm

5.2.1 PREPARATION phase

Suppose we already know the domain of discourse of this application denoted by $(\mathbb{C}, \mathbb{P})$. $\mathbb{C}$ is the set of *Classes* $(C_1, C_2, \ldots, C_n)$, and each C_i denotes one Class in the vocabulary that is "relevant" to the application. $\mathbb{P}$ is the set of *Properties* $(P_1, P_2, \ldots, P_m)$, and P_i is a "relevant". In Section 5.2.4 we will discuss further on how to determine this domain of discourse of an application given only query patterns.

Before doing filter expansion, we need to pre-process the ontology by substituting each of the anonymous classes with a unique class name. This can be easily done by using an ontology language parser. This is useful because it breaks down compound statements and simplifies the structure of ontology to a store of basic triples.

The filter expansion procedure is shown in Fig. 5.3. The input to the algorithm is the initial sets of $\mathbb{C}$ and $\mathbb{P}$. The output is the modified version of $\mathbb{C}$ and $\mathbb{P}$. $\mathbb{NC}$ is used in intermediate stages and $\mathbb{E}$ is kept in the description of algorithm only for consistent comparison in the following proof of completeness. The procedure is an iterative procedure, so the time complexity is hard to analyze. Nevertheless, we can estimate the worst case time complexity to be $O((N_C + N_P)^2 N_T)$, where N_C is the number of classes, N_P is the number of properties, N_T is the number of TBoxes in the ontology. The derivation of this complexity is as follows: The procedure contains a main loop with three nested loops. The main loop will loop for a maximum of $N_C + N_P$ times, assuming that $\mathbb{C}, \mathbb{NC}, \mathbb{P}$ would add only one element in each loop. The first and third

Let $\mathbb{NC}, \mathbb{E}$ be empty sets
add C, D in $\mathbb{C}$, add P in $\mathbb{P}$ *if* $D = (\leqslant nP.C)$
add C, D in $\mathbb{C}$, add P in $\mathbb{P}$ *if* $D = (= nP.C)$
add P in $\mathbb{P}$ *if* $\top \sqsubseteq \leqslant 1P.\top$
add P in $\mathbb{P}$ *if* $\top \sqsubseteq \leqslant 1P^{-}.\top$
loop until $\mathbb{C}, \mathbb{NC}, \mathbb{P}$ do not change
for all C_i *in* $\mathbb{C}$
    add D_1, D_2 in $\mathbb{C}$ *if* $C_i = D_1 \sqcup D_2$
    add D in $\mathbb{C}$, add C_2 in $\mathbb{NC}$ *if* $D = C_i \sqcup C_2$
    add D in $\mathbb{C}$ *if* $D = C_i \sqcap C_2$
    add D_1, D_2 in $\mathbb{C}$ *if* $C_i = D_1 \sqcap D_2$
    add D in $\mathbb{NC}$ *if* C_i is owl:complementOf D
    add $x_1, x_2, \ldots, x_l$ in $\mathbb{E}$ *if* $C_i = enum\{x_1, \ldots, x_l\}$
    add D in $\mathbb{C}$, add P in $\mathbb{P}$ *if* $D = \forall P.C_i$
    add D in $\mathbb{C}$, add P in $\mathbb{P}$ *if* $C_i = \exists P.D$
    add P in $\mathbb{P}$ *if* C_i owl:hasValue y on property P
    add D in $\mathbb{C}$ *if* C_i is owl:equivalentClass to D
    add D in $\mathbb{C}$ *if* $D \sqsubseteq C_i$
    add P in $\mathbb{P}$ *if* P rdfs:domain C_i
    add P in $\mathbb{P}$ *if* P rdfs:range C_i
end for
for all P *in* $\mathbb{P}$
    add C in $\mathbb{C}$ *if* C owl:hasValue y on property P
    add Q in $\mathbb{P}$ *if* $Q \sqsubseteq P$
    add Q in $\mathbb{P}$ *if* Q owl:equivalentProperty P
    add Q in $\mathbb{P}$ *if* Q owl:inverseOf P
end for
for all C_i *in* $\mathbb{NC}$
    add D in $\mathbb{NC}$ *if* $D = C_i \sqcup C_2$
    add D_1, D_2 in $\mathbb{NC}$ *if* $C_i = D_1 \sqcup D_2$
    add D_1, D_2 in $\mathbb{NC}$ *if* $C_i = D_1 \sqcap D_2$
    add C_2 in $\mathbb{C}$, add D in $\mathbb{NC}$ *if* $D = C_i \sqcap C_2$
    add D in $\mathbb{C}$ *if* C_i is owl:complementOf D
    add $x_1, x_2, \ldots, x_l$ in $\mathbb{NE}$ *if* $C_i = enum\{x_1, \ldots, x_l\}$
    add D in $\mathbb{NC}$ *if* C_i is owl:equivalentClass to D
    add D in $\mathbb{NC}$ *if* $C_i \sqsubseteq D$
    add D in $\mathbb{C}$ *if* C_i is owl:disjointWith D
end for
end loop

Figure 5.3: Filter Expansion

nested loops are complementary in the sense a class C can never be in both $\mathbb{C}$ and $\mathbb{NC}$. Therefore, the time required of these two nested loops is $O(N_C N_T)$, assuming that $\mathbb{C} + \mathbb{NC}$ covers all possible classes and inside each loop all TBox assertions are examined. Similarly, the complexity for the second nested loop is $O(N_P N_T)$. Combining together with the main loop, the worst case time complexity is $O((N_C + N_P)^2 N_T)$. Being polynomial, this complexity is significantly better than the worst case complexity of tableaux algorithm, which is exponential. The reason our algorithm is much faster is that we do not derive triples in the process. Rather, we simply use a filtering method to distinguish the useful from the useless, postponing the reasoning until queries are made. The space complexity of the algorithm is $O(N_C + N_P)$. The maximum number of filters equals the number of classes plus the number of properties. Other than the storage of filters, there is no significant space cost.

The filter expansion procedure is the soul of the whole algorithm. Essentially it is the specially expanded filter that breaks the monolithism, and makes distributed computing possible. In Section 5.3, we will prove the expansion procedure ensures the generated filters are complete for the type of query concerned. In addition, the proof also reveals that by taking the form of filtering, this expansion procedure produces the minimum set of filters to maintain completeness. In other words, the generated set of filters is both sufficient and necessary to answer queries completely.

5.2.2 SETUP phase

In SETUP phase we extract sub-ontologies from the base ontology. This includes both the vocabulary and the concrete information. Extracting the vocabulary for sub-ontology, if any TBox assertion in base ontology involves any element in set $\mathbb{C}$ or $\mathbb{P}$, this assertion is seen as a match and should be copied to sub-ontology. The filtering procedure for concrete information is carried out for all existing ABox triples in the knowledge base. Here we consider only 3 types of ABox assertions, leaving annotation assertions behind (Annotation assertions will not affect the semantic richness of sub-ontologies). The following notations are used: I_1, I_2, I are *individuals*, C is a *class* name, P is a *property* name.

1. $(I, \text{rdf:type}, C)$. If C falls in the set $\mathbb{C}$, copy the triple to sub-database.

2. (I_1, P, I_2). If the triple matches one element in set $\mathbb{P}$, copy it to sub-database.

3. $(I_1, \text{owl:sameAs}, I_2), (I_1, \text{owl:differentFrom}, I_2)$. All triples of these two types are copied.

5.2.3 UPDATE phase

When an update of triple is made to the knowledge base, it should be decided whether this update is relevant to an application, and therefore whether it should be delivered to the corresponding sub-database. This deciding procedure is the same as the filtering procedure in SETUP phase except that it is applied to a single triple instead of a set of triples.

5.2.4 Domain of Discourse

Now that we have finished describing the algorithm, we give a means to determine the domain of discourse $(\mathbb{C}, \mathbb{P})$ given the description of an application.

The domain of discourse directly determines the size of a sub-database. It should be designed in such a way that the yielded sub-database is big enough to hold all required information, and is small enough to exclude useless information. Since we distinguish sub-databases by different applications, application developers should be responsible for the definition of domain of discourse. Nonetheless, we give a guideline as follows:

1. First, enumerate all the possible query patterns that an application server might receive from its clients. This information can usually be taken from documentations from early application development phases.

2. Assume queries from clients are in the format of SPARQL or other query languages. If not in the case, abstract the interaction between application server and client as standard SPARQL queries.

3. In most queries, there is a *where* clause that may be constituted of multiple sub-clauses. If the predicate used in the sub-clauses is *rdf:type*, include the object (third element in a triple) of *rdf:type* in $\mathbb{C}$. If the predicate is a property name, include the predicate in $\mathbb{P}$.

Here we give an example of doing so. Suppose we have an application that has only one functionality, answering the user how much calories is contained in 100 grams of the asked food. From the documentation of use case

modelling of the application, we extracted the functionality and abstracted it as a SPARQL query: `SELECT ?x WHERE { thefood ns:hasCalories ?x. thefood rdf:type ns:Food }`, where `thefood` is the user input choice of recipe. From the *where* clause, we add *ns:hasCalories* to $\mathbb{P}$ and add *ns:Food* to $\mathbb{C}$.

5.3 Proof of Completeness

The completeness of an extraction algorithm is defined as such: Suppose the intact knowledge base is denoted by a set of triples $\mathbb{K}$. The output of the extraction algorithm is $\mathbb{S}$ ($\subset \mathbb{K}$). If all possible queries of the same syntax structure as the given type receive a same response from both $\mathbb{K}$ and $\mathbb{S}$, the extraction algorithm is said to be complete. Oppositely, if there exists a specific query received different responses from $\mathbb{S}$ and from $\mathbb{K}$, the algorithm is said to be incomplete. Because of the ability of inference inherent in ontologies, the proof is not as simple as it would be for relational databases. As an illustration, Fig. 5.4 shows the relationship between the existing triples, inferred triples and required triples. Existing triples $\mathbb{E}$ are triples that are already present in the database. With these triples, some more triples $\mathbb{I}$ can be inferred following the semantics of OWL language. The intersection between $\mathbb{E}$ and $\mathbb{I}$ indicates that some of the existing triples can be inferred by other existing triples. This is a sign of consistency of the knowledge base. For a specific type of queries, the set of triples $\mathbb{R}$ that is relevant can have overlap with both $\mathbb{E}$ and $\mathbb{I}$. With the Open World Assumption (OWA), some of the relevant triples may not even fall in the union of $\mathbb{E}$ and $\mathbb{I}$. A complete extraction of the database should at least comprise or

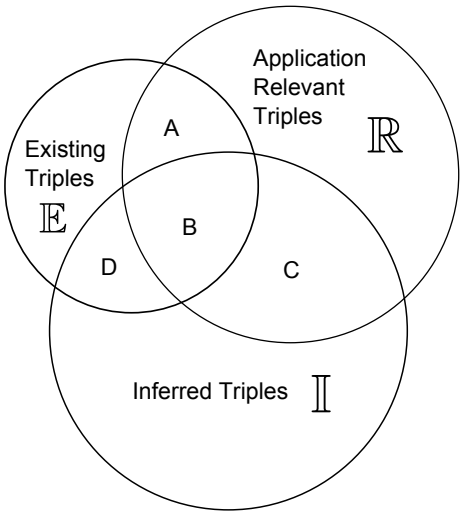

Figure 5.4: The whole knowledge base is $\mathbb{E}+\mathbb{I}$. A complete extraction should at least comprise $A+B$ and those that can be used to deduce C

be able to infer triples in part A, B and C.

Now that we understand that the objective of the algorithm is to make available part C as well as part A and B in sub-ontology, the challenge is how to find the triples in $\mathbb{E}$ that are used to infer triples in part C. To solve this problem, we need to understand every detail of the inferences in ontologies. The following two subsections are an exhaustive enumeration of deduction rules used in OWL knowledge bases.

According to [98], OWL is a vocabulary extension to RDF. Any RDF graph forms an *OWL Full* ontology. Further, the meaning given to an RDF graph by OWL includes the meaning given to the graph by RDF. OWL assigns additional meanings to certain RDF triples. Before analyzing the inference rules given by OWL, we first study inference rules given by RDF semantics.

Table 5.1: Selected Inference Rules in RDF Semantics

Rule Name	If E contains	Then add:
se1	uuu aaa xxx .	uuu aaa _:nnn . where _:nnn identifies a blank node allocated to xxx
rdf1	uuu aaa yyy .	aaa rdf:type rdf:Property
rdfs1	uuu aaa lll . where lll is a plain literal	_:nnn rdf:type rdfs:Literal . where _:nnn identifies a blank node allocated to lll
rdfs2	aaa rdfs:domain xxx . uuu aaa yyy .	uuu rdf:type xxx .
rdfs5	uuu rdfs:subPropertyOf vvv . vvv rdfs:subPropertyOf xxx .	uuu rdfs:subPropertyOf xxx .
rdfs7	aaa rdfs:subPropertyOf bbb . uuu aaa yyy .	uuu bbb yyy .
rdfD1	ddd rdf:type rdfs:Datatype . uuu aaa “sss”^^ddd .	_:nnn rdf:type ddd . where _:nnn identifies a blank node allocated to “sss”^^ddd .

5.3.1 Inference Rules Given by RDF Semantics

Inferences in RDF are usually called entailment. A full list of entailment rules in RDF semantics is defined in chapter 7 of [99]. This includes: 1. Simple entailment rules, 2. RDF entailment rules, 3. RDFS entailment rules, and 4. Datatype entailment rules. We chose a few representatives as listed in Table 5.1:

A closer look at the inference rules will reveal that many of them are not producing useful new triples. Simple entailments (type 1) and datatype entailment rules are on blank node generalization/instantiation. RDF entailment rules

are enforcing RDF syntax. Excluding these, only those inference rules given by RDF Schema (RDFS) can be used to generate new assertions. They can be further classified as class constraints (*rdfs:domain* and *rdfs:range*), class/property hierarchy (*rdfs:subClassOf* and *rdfs:subPropertyOf*).

5.3.2 Inference Rules Given by OWL Semantics

Unlike RDF inference rules, there is no readily available summary for OWL inference rules. In fact, an exhaustive enumeration of all OWL inference rules is very difficulty to produce. Luckily, as far as our application is concerned, we only need to focus on a proportion of all inference rules—those that can produce concrete class belonging relations and individual properties. We start with the summary of OWL DL axioms and facts at [100].

The rest of this section contains a lot of materials in Description Logic and First Order Predicate Logic. Readers can refers the [98, 100, 101, 23, 25] if find difficulty in reading.

An OWL ontology in the abstract syntax contains a sequence of annotations, axioms and facts. Annotations can be used to record authorship and other information associated with the ontology. The main content of an OWL ontology is carried in its axioms and facts. Axioms and facts are essentially TBoxes and ABoxes in terms of Description Logic, respectively. In other words, axioms and facts are the sources of extensions and restrictions given by OWL upon RDF. Inference rules can be extracted from the following 4 categories of statements: class constructors, class axioms, property axioms, and individual equivalencies.

Table 5.2: OWL Class Constructors

ID	Class Constructor	DL Syntax	FOL Syntax
1-1	owl:unionOf	$C_1 \sqcup \ldots \sqcup C_n$	$C_1(x) \vee \ldots \vee C_n(x)$
1-2	owl:intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	$C_1(x) \wedge \ldots \wedge C_n(x)$
1-3	owl:complementOf	$\neg C$	$\neg C(x)$
1-4	owl:oneOf	$\{i_1 \ldots i_n\}$	$x = i_1 \vee \ldots \vee x = i_n$
1-5	owl:allValuesFrom	$\forall P.C$	$\forall y.(P(x,y) \rightarrow C(y))$
1-6	owl:someValuesFrom	$\exists P.C$	$\exists y.(P(x,y) \wedge C(y))$
1-7	owl:hasValue	$\exists P.\{i\}$	$P(x,i)$
1-8	owl:minCardinality	$\geqslant nP.C$	$\exists y_1,\ldots,y_n.\bigwedge_{1\leqslant i\leqslant n}(P(x,y_i)\wedge C(y_i)) \wedge \bigwedge_{1\leqslant i<n, i<j\leqslant n} y_i \neq y_j$
1-9	owl:maxCardinality	$\leqslant nP.C$	$\forall y_1,\ldots,y_n.\bigwedge_{1\leqslant i\leqslant (n+1)}(P(x,y_i)\wedge C(y_i)) \rightarrow \bigvee_{1\leqslant i<(n+1), i<j\leqslant (n+1)} y_i = y_j$
1-10	owl:Cardinality	$= nP.C$	Conjunction of above two

Table 5.3: OWL Class Axioms

ID	Class Axiom	DL Syntax	FOL Syntax
2-1	owl:disjointWith	$C_1 \sqsubseteq \neg C_2$	$\forall x.\neg C_1(x) \vee \neg C_2(x)$
2-2	owl:equivalentClass	$C_1 \equiv C_2$	$\forall x.C_1(x) \Leftrightarrow C_2(x)$
2-3	rdfs:subClassOf	$C_1 \sqsubseteq C_2$	$\forall x.C_1(x) \rightarrow C_2(x)$

Table 5.2–Table 5.5 have shown all these structures with their Description Logic (DL) and First Order Logic (FOL) equivalents as an extension to the tables in [101].

An inference procedure can use one or multiple axioms listed in the tables. They are called primitive inference and composite inference, respectively. The final outcome of an inference procedure, however, can only have 2 forms: "$C(i)$", or "$P(i_1, i_2)$" despite the number of inference steps. Any other statements are only intermediate results and must be combined with other axioms to deduce useful results. For example, *owl:sameAs* can be used to deduce an equality formula $i_1 = i_2$, but equality formula usually cannot be used directly by applications. This formula, however, can be combined with $P(i_1, i_3)$ to produce $P(i_2, i_3)$.

The logic of the following enumeration is like this: We first list all possible primitive axioms and facts that will produce $C(i)$ and $P(i_1, i_2)$. The outcome of

Table 5.4: OWL Property Axioms

ID	Property Axiom	DL Syntax	FOL Syntax
3-1	rdfs:subPropertyOf	$P_1 \sqsubseteq P_2$	$\forall x, y.P_1(x, y) \rightarrow P_2(x, y)$
3-2	owl:equivalentProperty	$P_1 \equiv P_2$	$\forall x, y.P_1(x, y) \Leftrightarrow P_2(x, y)$
3-3	owl:inverseOf	$P_1 \equiv P_2^-$	$\forall x, y.P_1(x, y) \Leftrightarrow P_2(y, x)$
3-4	owl:SymmetricProperty	$P \equiv P^-$	$\forall x, y.P(x, y) \rightarrow P(y, x)$
3-5	owl:FunctionalProperty	$\top \sqsubseteq \leqslant 1P.\top$	$\forall x, y, z.(P(x, y) \wedge P(x, z)) \rightarrow y = z$
3-6	owl:InverseFunctionalProperty	$\top \sqsubseteq \leqslant 1P^-.\top$	$\forall x, y, z.(P(y, x) \wedge P(z, x)) \rightarrow y = z$
3-7	owl:TransitiveProperty	$P^+ \sqsubseteq P$	$\forall x, y, z.(P(x, y) \wedge P(x, z)) \rightarrow P(x, z)$
3-8	rdfs:domain	$\top \sqsubseteq \forall P^-.C$	$\forall x, y.P(x, y) \rightarrow C(x)$
3-9	rdfs:range	$\top \sqsubseteq \forall P.C$	$\forall x, y.P(x, y) \rightarrow C(y)$

Table 5.5: OWL Facts

ID	Fact	DL Syntax	FOL Syntax
4-1	owl:sameAs	$\{i_1\} \equiv \{i_2\}$	$i_1 = i_2$
4-2	owl:differentFrom	$\{i_1\} \sqsubseteq \neg\{i_2\}$	$i_1 \neq i_2$
4-3	i rdf:type C	$i : C$	$C(i)$
4-4	i_1 P i_2	$(i_1, i_2) : P$	$P(i_1, i_2)$

the inference is placed in the LHS, and the pre-conditions are put in the RHS, with facts placed after axioms. Among the terms of facts in the RHS of the rules, copy to LHS those that are absent in the LHS, and continue the process until no more updates to the listing. To kick start the procedure, we have the listing as Fig 5.5.

Copying those that are absent in LHS—$\neg C(x)$, and $x_1 = x_2$—to LHS, Fig 5.6 is yielded.

After this step, the only term on RHS that does not appear on LHS is $x_1 \neq x_2$, which shows inequality. However, inequality is not deductible. It can only be given in owl:differentFrom statements. So the procedure ends here with no more terms movable to LHS. Together we have 25 different forms of deduction rules.

$$
\begin{array}{rcl}
C(x) & \Leftarrow & C = D_1 \sqcup D_2, \quad D_1(x) \\
& \Leftarrow & D = C \sqcup C_2, \quad D(x), \neg C_2(x) \\
& \Leftarrow & D = C \sqcap C_2, \quad D(x) \\
& \Leftarrow & C = D_1 \sqcap D_2, \quad D_1(x), D_2(x) \\
& \Leftarrow & C = \neg D, \quad \neg D(x) \\
& \Leftarrow & C = \{x_1, \ldots, x_l\}, \quad x = x_i \\
& \Leftarrow & D = \forall P.C, \quad D(y), P(y, x) \\
& \Leftarrow & C = \exists P.D, \quad P(x, y), D(y) \\
& \Leftarrow & C = \exists P.\{i\}, \quad P(x, i) \\
& \Leftarrow & C \equiv D, \quad D(x) \\
& \Leftarrow & D \sqsubseteq C, \quad D(x) \\
& \Leftarrow & \forall x, y.P(x, y) \rightarrow C(x), \quad P(x_i, y) \\
& \Leftarrow & \forall x, y.P(x, y) \rightarrow C(y), \quad P(y, x_i) \\
& \Leftarrow & x = y, \quad C(y) \\
P(x, y) & \Leftarrow & C = \exists P.\{y\}, \quad C(x) \\
& \Leftarrow & Q \sqsubseteq P, \quad Q(x, y) \\
& \Leftarrow & P \equiv Q, \quad Q(x, y) \\
& \Leftarrow & P \equiv Q^-, \quad Q(y, x) \\
& \Leftarrow & P \equiv P^-, \quad P(y, x) \\
& \Leftarrow & P^+ \sqsubseteq P, \quad P(x, z), P(z, y) \\
& \Leftarrow & x = z, \quad P(z, y) \\
& \Leftarrow & y = z, \quad P(x, z)
\end{array}
$$

Figure 5.5: Primitive Inference Rules

$$
\begin{array}{rcl}
\neg C(x) & \Leftarrow & D = C \sqcup C_2, \quad \neg D(x) \\
& \Leftarrow & C = D_1 \sqcup D_2, \quad \neg D_1(x), \neg D_2(x) \\
& \Leftarrow & C = D_1 \sqcap D_2, \quad \neg D_1(x) \\
& \Leftarrow & D = C \sqcap C_2, \quad C_2(x), \neg D(x) \\
& \Leftarrow & C = \neg D, \quad D(x) \\
& \Leftarrow & C = \{x_1, \ldots, x_l\}, \quad x \neq x_i, 1 \leqslant i \leqslant l \\
& \Leftarrow & C \equiv D, \quad \neg D(x) \\
& \Leftarrow & C \sqsubseteq D, \quad \neg D(x) \\
& \Leftarrow & C \sqsubseteq \neg D, \quad D(x) \\
x_1 = x_2 & \Leftarrow & D = (\leqslant 1P.C), \\
& & D(y), P(y, x_1), C(x_1), P(y, x_2), C(x_2) \\
& \Leftarrow & D = (\leqslant nP.C), \\
& & D(y), \bigwedge_{1 \leqslant i \leqslant (n+1)} (P(y, x_i) \wedge C(x_i)), \\
& & \bigwedge_{1 \leqslant i < (n+1), i < j \leqslant (n+1), i+j-3 > 0} x_i \neq x_j \\
& \Leftarrow & D = (= nP.C), \\
& & D(y), \bigwedge_{1 \leqslant i \leqslant (n+1)} (P(y, x_i) \wedge C(x_i)), \\
& & \bigwedge_{1 \leqslant i < (n+1), i < j \leqslant (n+1), i+j-3 > 0} x_i \neq x_j \\
& \Leftarrow & \top \sqsubseteq \leqslant 1P.\top, \quad P(y, x_1), P(y, x_2) \\
& \Leftarrow & \top \sqsubseteq \leqslant 1P^-.\top, \quad P(x_1, y), P(x_2, y) \\
& \Leftarrow & x_1 = x_3, x_2 = x_3
\end{array}
$$

Figure 5.6: Primitive Inference Rules (Cont.d)

5.3.3 Proof of Completeness

After giving an exhaustive enumeration of useful inference rules (inferences rules that eventually can deduce $C(x)$ or $P(x, y)$), now we proceed to prove that any triples in part C in Fig. 5.4 will be deducible in our sub-ontology.

For the simplicity of demonstration, we assume $\mathbb{C} = \{C_T\}, \mathbb{P} = \{P_T\}$. The proof is done by contradiction.

Suppose there is at least one triple in part C that is not deducible by the sub-database using our algorithm, denoted by $triple_C$. It can be format of either $C_T(x_0)$ or $P_T(x_0, y_0)$. In either case, we can construct an inference tree for the triple in the context of the whole knowledge base. The root node of the tree is the $triple_C$. At each branching, the children nodes are the axioms or facts that are used to deduce their parent node. Each deduction is atomic and cannot be further divided. In other words, each deduction corresponds to a primitive inference rule in Fig 5.5 or Fig 5.6. If $triple_C$ can be deduced from multiple approaches, we simply choose one of them to form the inference tree. The inference tree keeps expanding until all leaf nodes are existing axioms or facts that require no further deduction. So altogether, the tree manifests the inference procedure in producing $triple_C$ from scratch. The inference tree will have such features: Each branching corresponds to an inference rule in Fig 5.5 or Fig 5.6; At each branching, the parent node is a fact and at least one of the children is another fact.

Because $triple_C$ cannot be deduced in sub-database (as assumed), there must be some leaf node presented in the inference tree that is absent from the extracted

sub-database. Consider all possible cases:

1. The leaf node denotes a fact of equality or inequality and it is not in the extracted sub-database. However, all *owl:sameAs* and *owl:differentFrom* statements are copied to sub-database without going through the filters. This yields contradiction.

2. The leaf node denotes a fact $C_m(x_m)$. Hence, C_m is not included in the set $\mathbb{C}$ in PREPARATION phase. Traversing from this leaf node back to the root node, we will come across a sequence of facts $f_1, f_2, \ldots, f_{n-1}, f_n$, where f_1 is $C_m(x_m)$ and f_n is $triple_C$. Because $C_m(x_m)$ does not fit in our algorithm's filters but $triple_C$ does, along the sequence we can find the first fact f_i fits one of the filters while f_{i-1} does not. The inference rule $f_i \Leftarrow a_{i-1}, f_{i-1}$ shall be one of the rules in Fig 5.5 and Fig 5.6. Now that there is a bijection between the filter expanding procedure and the rules in Fig 5.5, the filter that can match f_i is sure to be expanded to a filter that can match f_{i-1}. This contradicts with the knowledge that f_{i-1} does not fit in any filters in the algorithm.

3. The leaf node denotes a fact $P_m(x_m, y_m)$. The same logic as case 2 will yield contradiction.

4. The leaf node denotes an axiom. Because an axiom can never be the only child of a parent node, it must have a sibling denoting a fact. We have just now proved a fact node in inference tree is always included in sub-ontology. Discerning the axiom part and the fact part of a primitive

inference rule is always directly related, this axiom node is surely to be included in sub-ontology according to the procedure of our algorithm.

In all 4 possible cases, we will have contradictions. Thus, the initial supposition is false, and our algorithm is complete. □

5.4 Trade Completeness for Lightweight Sub-databases

When implementing the algorithm, it is observed that the domain of discourse usually expands from the initial one or several classes and properties to tens of them. This is because of the convoluted semantics in ontology knowledge bases. Strictly speaking, removing any one of the filters will always render a loss of completeness in query answering. However practically, we are usually facing a more "friendly"-designed knowledge base where the inherent consistency of knowledge base grants us the chance to prune some of the filters without harming the performance. It is worth noting that pruning some of the filters does undermine the completeness. In the cases when the updates to the knowledge base are not as self-contained as the current level, erroneous query-answering can appear. In other words, we trade some of the overshooting completeness for actual benefits—smaller sub-databases, with the assumption that the level of self-consistency is to be remained in the future.

This section studies how much completeness we can sacrifice and how much we can gain from a smaller sub-database, so as to decide whether such trading is instrumental.

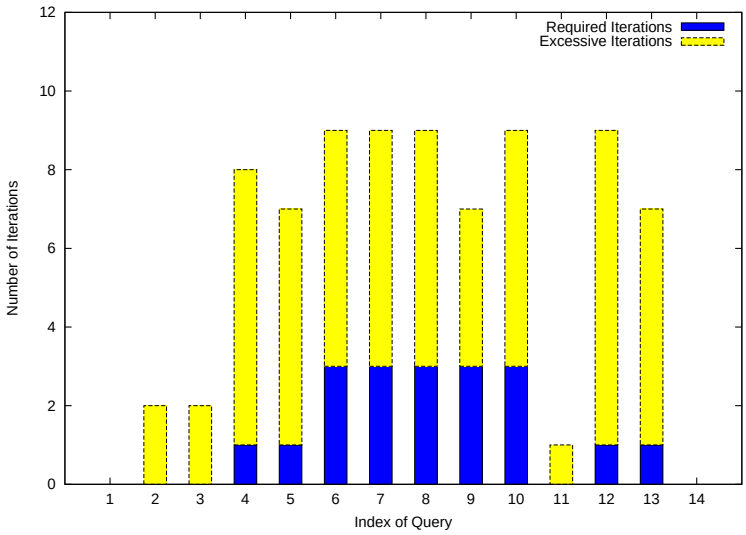

Figure 5.7: Proportions of Required and Excessive Iterations

A test of our algorithm on LUBM[95] is employed emphasizing on how the number of iterations impact the outcome. As shown in Fig 5.7, the height of each stacked bar denotes the number of iterations that must be performed before our algorithm converges. For query 1 and 14, the height is 0 indicating the domain of discourse is converged at the beginning. Similarly, 9 iterations are required for query 6,7,8,10,12 to converge. If we do not wait until the convergence, the filter set is not guaranteed to be complete, and thus the reply to queries might be fragmentary. Due to the fact that knowledge bases are usually self-consistent in many aspects, some assertions that are explicitly presented can be deduced from other assertions implicitly. In such occasions, accounting for these self-contained inferences will not add to the semantic richness of the sub-database, but will only produce a larger and more cumbersome sub-database. In the case of LUBM, the algorithm requires at most 3 iterations to produce a practically complete set of filters for all query types.

From Fig 5.7 we also learn the number of excessive iterations are usually a multiple of the number of required iterations. How much exactly we can benefit by exploiting this observation? Fig 5.8 shows our attempt. Assuming we have the knowledge of required iterations and excessive iterations beforehand, we limit filter expansion procedure to be run for only required iterations and then measure the size of sub-databases. This size is then compared against the size when a full procedure is carried out until convergence. Fig 5.8 shows the proportional size reduction for all query types. Except for query type 1, 3, 11, and 14, other query types experience different level of storage reduction. The zero reduction for type 1 and type 14 is as anticipated because their required iteration is the same as total iteration—both were zero. However, for type 3 and type 11, though there are non-zero excessive iterations, these iterations did not bring in concrete effect in terms of ontology size. Similarly, large numbers of excessive iterations also do not imply great reduction in sub-database size. It is possible the few required iterations have already incorporated a large number of individuals and triples. The reduction is most evident for query type 12, where a 99% reduction in both the number of individuals and the number of triples is observed.

The next problem is how to determine which iterations are required, and which are excessive. Resolving this problem using analytical methods is very hard, if even possible. This is because the number of required iterations is dependent on the feature of the current database. Remember the whole trade-off concept is built upon the assumption that future updates to the database will maintain current self-consistency level. To approach this problem analytically,

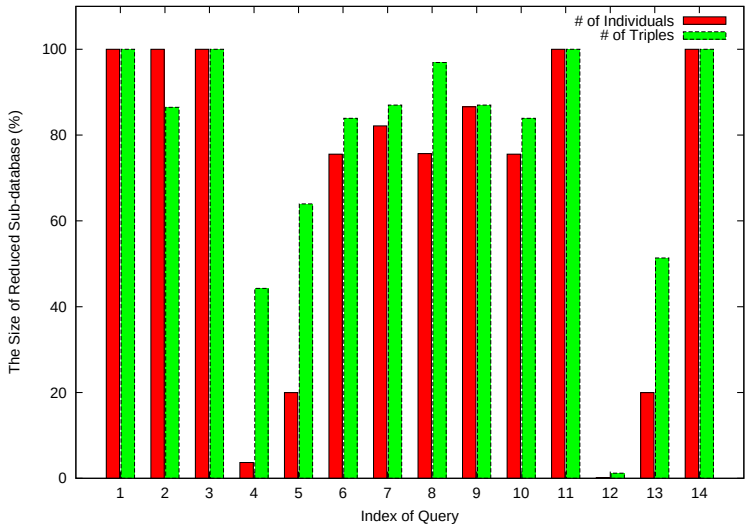

Figure 5.8: The Size of Sub-databases if Excessive Iterations are Removed (Presented as a Percentage of the Full version)

one need to extract all the self-consistency features of the database before any analysis. This is considered even harder than building such a database from scratch. So we suggest developers to determine the number of required iterations by building a prototype system. By decreasing the number of iterations used from maximum value (the number of iterations to achieve convergence) to zero, developers should constantly monitor if the query results are different from the results given by a complete algorithm. Whenever a different result is present, the empirical value of the number of required iterations is found. Usually the level of self-consistency can be fluctuating in a small range, so this number can be adjusted accordingly. When the level of self-consistency improves and the sub-database has grown too big, one can reduce the number of iterations used. When conflicts or incomplete query answers are detected, one should adjust the number upwards.

5.5 Evaluation

5.5.1 Performance Evaluation

We evaluate our algorithm on the test bench proposed in Chapter 4. There are two pairs of comparison in this evaluation. In logical sequence, the first pair is comparing the performance of our algorithm with a trivial solution—server cloning. The second one is comparing the performance of our algorithm with any other reasoning-based algorithm that achieve distributed computing scenario. However, with the data measured in the second comparison, the outcome of the first comparison would be obvious. So we deliberately consider the second comparison first.

Our algorithm has its strength in the running speed of triple synchronization. A reasoning-based algorithm will need to go through an OWL reasoning procedure and a query-answering procedure before deciding whether a triple is "relevant" to a sub-database. The faster running speed is achieved at a cost. We also need to evaluate how much cost is incurred. The cost of our algorithm is that we will probably need to construct a sub-database larger than necessary because of the extra triples that are useful for deducing relevant information.

The system is built upon *Jena*[102], with an extensional reasoning support from *Pellet*[103]. It uses TDB as a persistent storage of triples. The system is developed on Java 1.6 and Eclipse Indigo, tested on Windows 7 Professional, Core 2 Duo E8500, 3.16GHz.

Since we are comparing our algorithm with no specific reasoning-based algo-

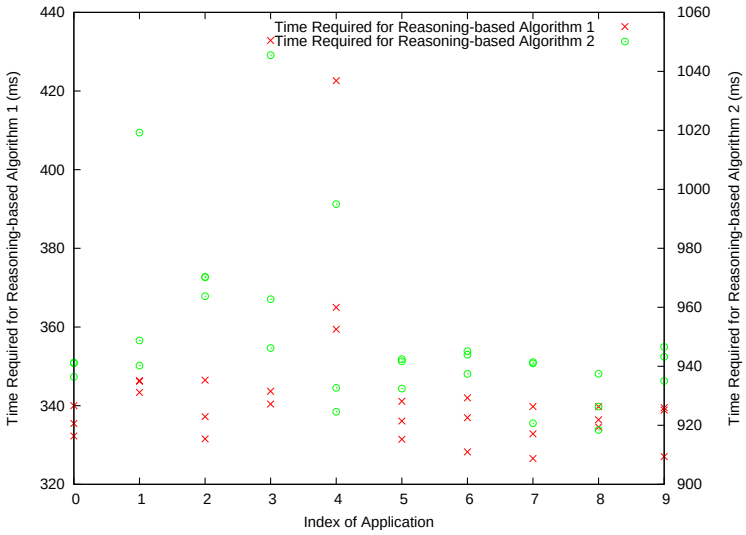

Figure 5.9: Reasoning-based algorithm requires hundreds of milliseconds

rithm, we use the running speed of some key procedures to represent the minimum required running time of any algorithm that stems from OWL reasoning and query-answering. These key procedures are abstracted as reasoning-based algorithm in following texts, and our algorithm is compared with this "reasoning-based" algorithm.

Fig 5.9 shows the time required for a re-classification and a query-answering in reasoning-based algorithms. In order to fully utilize the caches in ontology management software, we first perform a classification for the knowledge base and fire the query once. Then after a minor change to the knowledge base, we measure the processing time for a re-classification and a query-answering that immediately following the change. Whether the minor change is to insert or to remove a triple can impact the performance much, so both of them are listed in Fig 5.9 as *Algorithm 1* and *Algorithm 2*, respectively. However, altering the triple to be inserted or removed made no significant impact on the running

speed according to multiple comparison experiments. In Fig 5.9 the experiment is carried out for 10 categories of applications and each application has 3 types of queries. The average processing time of 17 queries from a same type is plotted on the graph as a dot. The re-classification and query-answering after insertion and removal will take around 350 ms and 950 ms respectively. Referring to [32, 33, 35, 37], the running time using different reasoner is still roughly in the same order of magnitude despite one algorithm may perform better than another. Therefore, in our experiment, we can conclude that with a dataset on the domain of context-awareness and of similar size of ours, any reasoning-based solution for triple synchronization will take hundreds of milliseconds to complete.

On the other hand, it is tested that our algorithm for triple synchronization requires less than 1 ms. This absolutely is a drastic improvement that eases the burden on central server when an update triple arrives. This is a natural result as the filtering process takes only hundreds of machine cycles to complete while a reasoning process is much more complicated. It is worth noting here again, that this drastic improvement is achieved essentially by postponing the reasoning until really necessary. But without all the analysis and proofs in our work, there is no way to postpone it. The reasoning has to be done on the spot, thus dragging down the performance of the whole system as a bottleneck. The benefit of our algorithm is conspicuous, next we should discuss the cost of our algorithm, i.e., how much extra storage space is required to make this algorithm possible.

Table 5.6 gives an overview of the storage required for both a reasoning-based

Table 5.6: Storage Cost of the Algorithm

Reasoning-based Algorithm	#0	#1	#2	#3	#4	#5	#6	#7	#8	#9
Individual Count	1201	688	953	799	923	529	344	681	2042	541
Triple Count	7207	2093	4506	1713	3317	1390	1136	1713	2836	724
Proposed Algorithm	#0	#1	#2	#3	#4	#5	#6	#7	#8	#9
Individual Count	1201	688	953	799	808	529	344	681	1863	541
Triple Count	27478	31953	22857	22665	23502	22527	21929	20718	23638	23335

and our proposed algorithm. Mostly the triples included in proposed algorithm is greatly larger than the one in reasoning-based solution. The increase ranges from 4 times to 15 times in our test bench. This also gives us a hint on the relative size of parts in Fig 5.4. One might notice that sometimes the individual included in proposed algorithm is even smaller than that in reasoning-based solution. This is because the reasoning-based solution will explicitly incorporates some deduced individuals while they are left to be deduced in the proposed algorithm.

Comparing the gain and costs of the proposed algorithm, we observe that the time required for triple synchronization drops from several hundreds of milliseconds to less than 1 ms, at the cost of a 10 times increase in storage space. At a time when mass storage is much cheaper than processing unit, we can assert that introducing such algorithm to distributed ontology computing can be beneficial in total.

We also tested our algorithm on LUBM benchmark for further justification. Using random seed zero, an ontology for an university of 15 departments is constructed. This ontology has 43 classes, 25 object properties, 7 data properties, 17174 individuals, 49336 *ObjectProperty* assertions, and 33079 *DatatypeProperty* assertions.

Not surprisingly, the triple synchronization time of our algorithm is still below

Table 5.7: Performance of Reasoning-based Algorithm on LUBM

Query	Algorithm 1 (ms)	Algorithm 2 (ms)
1	328	615
2	> 1hour	> 1hour
3	341	656
4	247	525
5	363	681
6	323	650
7	250981	252944
8	61533	61575
9	> 1hour	> 1hour
10	390	739
11	219	555
12	310	658
13	353	670
14	570	590

1 ms. In contrast, the running time for reasoning-based algorithm is shown in Table 5.7. Query 2 and query 9 are so complicated that their processing time exceeds 1 hour. Similarly, query 7 and query 8 also requires a significantly longer time to process as compared to other queries because of higher complexity in their query pattern. Despite these anomalies, the average processing time after insertion and that after removal are 350 ms and 650 ms, respectively.

As for the cost of the algorithm, they are studied in Fig 5.10 and Fig 5.11. In addition to the proposed algorithm and the reasoning-based algorithm, the tailored algorithm (with trade-off) that we discussed in Section 5.4 is also studied. The two figures are prepared using the size under proposed algorithm as a normalizing factor, showing the relative size of tailored algorithm and reasoning-based algorithm. Results show that the size of a reasoning-based sub-database is usually a percentage to the size of our proposed algorithm. In a few cases, this ratio drops below 10%, but mostly it is around 40%-80%. This implies the storage

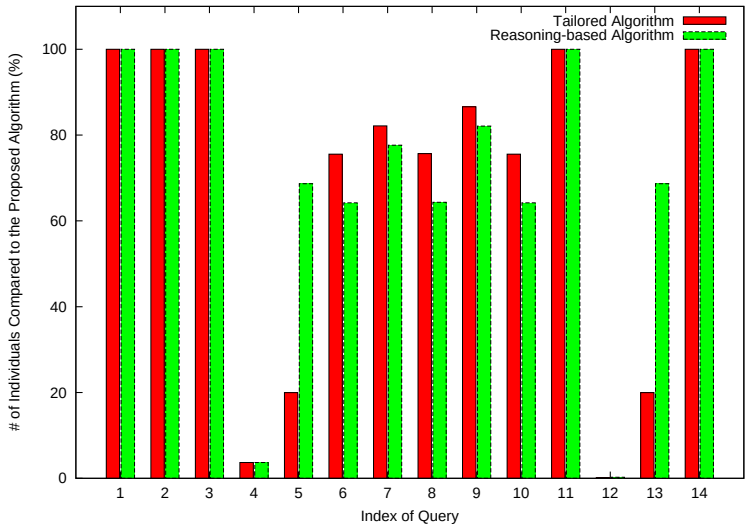

Figure 5.10: Storage Cost on LUBM (Number of Individuals Included)

cost of our proposed algorithm is usually 2-3 times, and occasionally can reach as high as 50 times of the size required by reasoning-based algorithms. This cost when compared to the running speed improvement is acceptable. After all, the size of sub-database can never be bigger than the whole knowledge base, putting an upper bound on the size of sub-databases. There is an abnormal case that for query 11, the number of triples for reasoning-based algorithm is even bigger than that of our proposed algorithm. This is because the reasoning-based algorithm explicitly incorporated some deduced triples, while in our algorithm, they are not included. Another observation is that the size of sub-database under tailored algorithm is smaller and closer to the number required by the reasoning-based algorithm. This means the cost can be further reduced leveraging the trade-off between completeness and lightweightness.

Recall at the beginning of Section 5.5.1, we mentioned a complete evaluation include two comparisons. Now it is time we consider the other comparison—

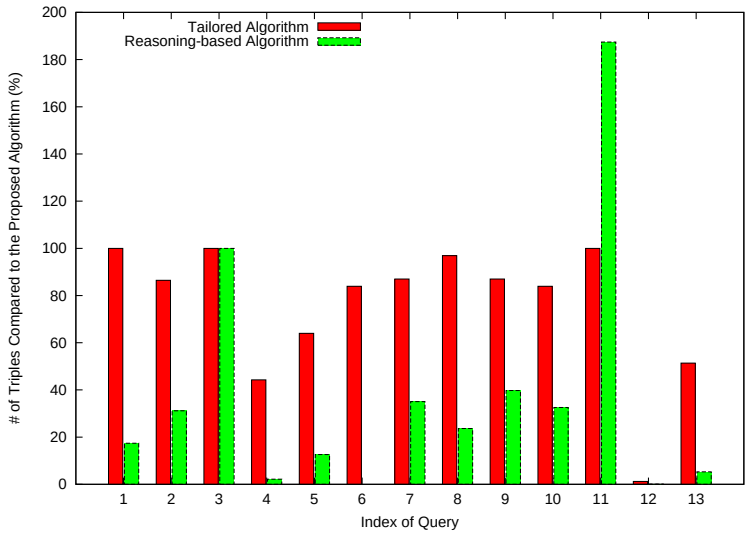

Figure 5.11: Storage Cost on LUBM (Number Of Triples Included)

between our algorithm and the clone server method. In terms of synchronization speed, the clone server method has the fastest speed. Clone server method requires literally no time to do synchronization. All updates are pushed to all other servers without any computation or decision. As discussed previously, our algorithm still requires a filter matching process for the synchronization. However, this process is fast enough (within 1 ms) so it would not be a problem. Secondly, in terms of the size of sub-databases, our algorithm is better than clone server method as our sub-databases are only sub-sets of the whole knowledge base. A quantitative evaluation is shown in Fig 5.12 and Fig 5.13. Results show sub-databases usually contain only 1%–6% individual counts and 30%–40% triple counts. Both of them are significantly below 100%. There is another important factor that makes clone server method much worse—the amount of synchronization traffic. Clone server method requires all updates to be pushed to other servers, resulting enormouse amount of traffic. In our algorithm however,

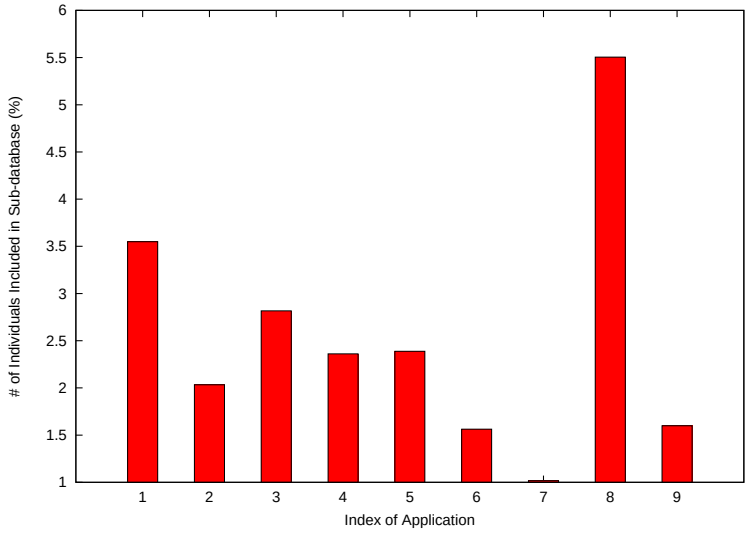

Figure 5.12: Size of Sub-databases (Individual Count as Percentage to the Whole Database)

most individual updates need no or a single forward to other servers, and most triple updates need only 2-3 forwards to other servers. Note there is an exception in our approach that an individual update of class Person is required to forward to all other sub-databases. This happens when a new user joins the system. So this type of expensive synchronization is capped at the size of user group. To sum up, our algorithm provides better performance in terms of the size of sub-databases and the amount of synchronization traffic. The cost here is the less than 1 ms synchronization time, which is acceptable.

5.6 Chapter Summary

In this chapter, we proposed a fast and complete algorithm to extract sub-ontologies from a base ontology for a given task, and also to keep the sub-ontology updated whenever changes are issued to the base ontology. The key

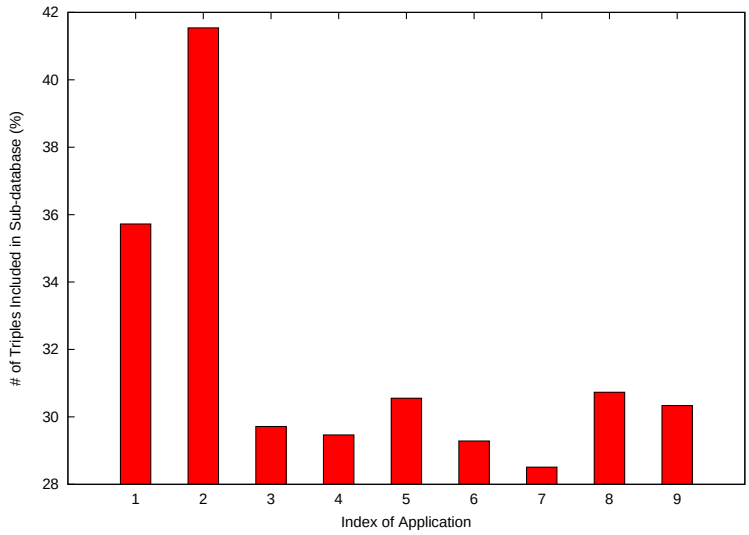

Figure 5.13: Size of Sub-databases (Triple Count as Percentage to the Whole Database)

contribution of this algorithm is the idea of filter expansion. The prolonged OWL reasoning procedure is replaced by this filter expansion algorithm at the time doing synchronization decision. The reasoning process is postponed until really necessary (when a query is received). Instead of using a large amount of time in TBox classification, the filter expansion procedure focuses on ABox selection. Starting from a small set of classes and properties, an iterative method is employed to expand the domain of discourse to a larger set. This larger set of filters is necessary because otherwise the query-answering may be incomplete. When filtering the ABox of the knowledge base by using this filter set, the triples that are directly useful for the query as well as the triples that can derive useful information are passed to the sub-database. This seemingly simple replacement nevertheless drastically reduces the processing time.

When we see the thousandfold improvement in processing speed, the cost of

the algorithm is measured to be tens of times larger storage. While this result is by itself acceptable already, we can further reduce the cost by applying a trade-off, exchanging some overshooting completeness for the benefit of smaller sub-databases. By restricting the maximum number of iterations in filter expansion, the algorithm can be suspended prematurely. Though there is the risk of future data structure changes, this technique will result in a smaller domain of discourse, a smaller sub-database, and lower deployment costs.

Chapter 6

Context-aware Recommendation System

6.1 Introduction

Mobile commerce has emerged as the ubiquity of smart phones and ultra fast mobile data network. While much of the work done in the field of mobile commerce has focused on the customer behaviour, business model, and wireless infrastructures, this work proposes a context-aware solution to mobile commerce. Context-awareness can improve customer's shopping experience, and thus is crucial in advocating mobile commerce. In this section, an entertainment recommendation system is described. Special treatment of context information storage and its usage in recommendation systems are analyzed. The context-aware collaborative filtering algorithm (CCF) proposed in the work is tested to produce better performance as compared to a traditional context-enabled CF method. A

prototype of the system is built on the domain of music recommendation and it is well received by selected users.

According to IDC Financial Insights' 2012 Consumer Payments Survey, 34 percent of survey respondents have made a purchase using their mobile phone compared to 19 percent a year ago. This clearly shows the emergence of mobile commerce. The report also found that physical goods were the most common mobile purchase, with more than 70 percent having purchased a physical good. 60 percent have purchased online services and digital goods instead.

Many technical aspects of mobile commerce research have reached commercializing quality. The works that remain are largely to steer the shift of user purchase habit from stationary computers to mobile terminals, and to improve users' experience when doing such purchases to accelerate this shift. Context-aware applications, being first introduced to improve user experience, are the natural choice for further advocating mobile commerce.

However, existing research on mobile commerce mostly solves problems like behaviour, business model, wireless infrastructure, etc. How to take context information into account is seldom considered. This work mainly looks into two aspects of incorporating context into mobile commerce: How to capture and represent context information, and how the context information is used to provide better services. The "services" mentioned here specifically mean recommending users to the most relevant product in the current situation.

The scenario in this work is set as follows: Alice is on her way back home after one day's work. The subway to home is taking approximately 1 hour so

she decided to have some fun on the phone to kill time. When she refers to our application, the system detects Alice has following contexts: She is leading a frugal life style, though purchasing is OK, it is limited to 1 dollar (from previous purchase history or inputted personal profile); She is on her way back home and the time is estimated to be 1 hour (from current GPS position and pre-defined home/work location); She has got no company so the entertainment should be for a single person (from the observation that no friend's device is nearby); Though the earplug is on, ambient noise level is high (from earplug detection and speaker sampling); Alice is fond of reading verse, listening to symphonies and playing social network games (from personal profile).

An ideal system should make an overall evaluation of the user's current situation, and finally comes to the decision to recommend a little social network game Alice's friends are playing online. But building such a system from the expert system approach is difficult. The sheer number of recommendation rules can overburden the developers. In this work, we propose to build such a system using Collaborative Filtering (CF) techniques that are specially tuned to consider context information. Using CF techniques, we assume Alice choice will be similar to like-minded users' choices, so no explicit rules are required.

The contribution of this work includes:

- We proposed a novel system that captures and manages context information to be used by recommendation algorithms.
- We have formulated a context-aware collaborative filtering algorithm (CCF). This algorithm managed to solve the recommendation problem in 2-D space

instead of higher. Context information is considered quantitatively rather than qualitatively.

This chapter is organized as such: Section 6.2 focuses on the context information gathering and distribution. The context-aware collaborative filtering algorithm is explained in detail in Section 6.3. Section 6.4 evaluates both the algorithm and the prototype system and we conclude this chapter in Section 6.5.

6.2 System Overview

In this section we explain details of the proposed system, but leave the recommendation algorithm in next section.

From the perspective of data structure, this system comprises two databases. The first one is the rating table, storing all user ratings as well as user contexts. This is a 2-D table with rows corresponding to users and columns corresponding to items/contexts. The second database is ontology-based, storing the profiles of users, properties of the items as well as their relationships. To link these two databases, a translation table is constructed to translate ontology objects to row/column number and vice versa. The ontology-based database is constructed because of ontology's power to do reasoning. When the rating table is sparse, meaning not many people have rated items, we can depend on the ontology reasoning to provide satisfying results.

Context information gathered can be either physical sensor data, or user profile. The user profile can be obtained through user inputs, data crawling on the

user's related homepages, or implicitly derived from the user's past behaviours.

Modern smart phones have evolved into something that is much more than a telecommunication tool. Many sensors are embedded in the smart phones, and this number is still increasing. Smart phone sensors can detect ambient noise level, ambient light level, moving speed, turbulence level, GPS location, etc. These sensor data can be used to reproduce the physical environment of the user in the virtual world. The richness of user context enables high quality of artificial intelligence in our system. For example, in a typical E-commerce system, the user may be recommended a serene verse if the user had shown preference in tranquil readings. In a context-aware mobile-commerce system, we may be able to do better than that. If the user is detected to be in a noisy and trembling subway cabinet where focusing on a quiet reading material is difficult, it is probably better to recommend some music for the user in the genre she had shown interest in.

Besides the physical context of the user, the profile of the user can also be detected. With the user's permission, our system can gain access to the user's social network sites and screen her friend list and/or historical posts. Using information retrieval techniques, this inspection can provide the recommender system with more knowledge about the user's preferences. User preferences together with user's purchase history in E-commerce sites constitute the context of this user in a longer time frame when compared with the context gained by sensor data interpretation. Moreover, statistics show that most people's purchase decisions are suggested by friends, because friends' recommendation has the

highest trust level. The area of social commerce has been investigating this phenomenon for long. Our system embraces this feature whenever it is possible to retrieve a friend's profile.

6.3 Recommendation Algorithm

In this section we describe how we integrate the context information in the recommendation process.

Our system uses a modified item-based collaborative filtering algorithm for giving predictions and recommendations, called Context-aware Collaborative Filtering (CCF). Unlike MD[104] and RST[105], our approach managed to limit the dimensionality of the algorithm in 2-D space.

6.3.1 Context-aware Collaborative Filtering

The goal of a CF algorithm is to predict the utility of a certain item for a particular user based on the preference (both explicit and implicit) of the target user and other like-minded users. Usually an $m \times n$ rating matrix $\mathbf{R}$ is employed to represent all the user-item data. Each entry of the matrix $R_{i,j}$ in $\mathbf{R}$ represents the rating of the ith user on the jth item. Ratings are in a numerical scale indicating the preference of the user. A typical application uses 1 to 5 to denote lowest preference to highest preference, and 0 is used to represent that item is not yet rated by the user.

A multi-dimensional rating matrix that incorporates context information is denoted by $\mathbf{R}_{m \times n \times c}$, where c is the number of context information types. CCF

unfolds the third dimension and has a rating matrix denoted by $\mathbf{R}_{m\prime\times(n+c)}$, where $m\prime$ is the unfolded user number. For example, a user may have rated on two movies under different context sets. By duplicating the user, these two ratings are projected to the 2-D space as 2 rows. The first/second row represents the rating and contexts for the first/second movie, both given by the same user.

In a traditional context-aware collaborative filtering algorithm, the third dimension (context information) is simply used as a filter condition. Only if the value of context information exceeds some arbitrarily-set threshold, the rating as a whole can be considered in later predictions. It is hard to determine the threshold and this scheme restricts the usage of context information. In our approach, context information is treated as a special class of items. The columns of the matrix can be divided into *normal items* and *context items*. The meaning of the *ratings* in this matrix is also augmented, and is thus referred to as extended ratings. With extended ratings, the kernel of CF algorithm can remain largely intact while considering the effect of context information. Other modifications of CF algorithm involve a *quantification of context information*, and an extra *weighting scheme*.

Quantification of context information converts context information of various formats into the same format as ratings. Context information that is on a continuous scale is quantified to its nearest integer between 1 and 5. Binary or Boolean context information is either grounded to 1 or raised to 5. Some of the context information may have discrete value and the size of the range is greater than 5. They are restructured as a series of ratings, or more mathematically, a

vector of the same length as the size of its range. In this vector, the field that corresponds to the currently active value will be set to 5 and all others are set to 1.

The weighting scheme in our approach endows extra significance to context items than normal items. In typical CF algorithms, the last step is usually a weighted sum prediction:

$$P_{u,i} = \frac{\sum_{N_j \in \mathbf{N}} (s_{i,N_j} * R_{u,N_j})}{\sum_{N_j \in \mathbf{N}} (|s_{i,N_j}|)} \tag{6.1}$$

where $P_{u,i}$ is the prediction of user u's opinion towards target item i, $\mathbf{N}$ is the set of items that are similar to item i, s_{i,N_j} is the similarity between item i and item N_j, and R_{u,N_j} is the user u's rating on item N_j. The similarity measure used in our system is correlation-based:

$$s_{i,j} = \frac{\Sigma_{u \in \mathbf{U}}(R_{u,i} - \overline{R_i})(R_{u,j} - \overline{R_j})}{\sqrt{\Sigma_{u \in \mathbf{U}}(R_{u,i} - \overline{R_i})^2}\sqrt{\Sigma_{u \in \mathbf{U}}(R_{u,j} - \overline{R_j})^2}} \tag{6.2}$$

where $\mathbf{U}$ is the set of users who both rated item i and item j.

Equation 6.1 is a weighted sum of ratings, which is then normalized to scale. With the introduction of context information, the predicted rating should be defined in a way that context information is also considered. To account for the effect of different context information, we assign a vector of weights in the

modified prediction:

$$P_{u,i} = \frac{\sum\limits_{N_j \in \mathbf{N}} s_{i,N_j} * R_{u,N_j} + \sum\limits_{C_j \in \mathbf{C}} w_{u,C_j} * s_{i,C_j} * R_{u,C_j}}{\sum\limits_{N_j \in \mathbf{N}} (|s_{i,N}|) + \sum\limits_{C_j \in \mathbf{C}} (|w_{u,C_j} * s_{i,C_j}|)} \tag{6.3}$$

where $\mathbf{C}$ is the set of context items. Note that R_{u,N_j} can be read from the multiple rows in the rating matrix because there are multiple rows that correspond to the user u. If the predicted rating happens to be beyond the range of common ratings, it is capped or grounded to the limits. The parameter w_{u,C_j} is determined by a learning process. It represents the level of fastidiousness of user u on context C_j. When it is set to 0, it means the context item is completely irrelevant to this user. When given a value greater than 1, the context item is treated with escalated importance. Note that this set of parameters are independent of the target item i, it is only dependent to user and context item. The effect of changes of target item is completely embodied in the parameter s_{i,C_j}.

6.3.2 Learning Process

The system computes the similarities between pairs of normal items, between pairs of normal item and context item, before any user's visit. In addition to that, the weight parameters introduced in CCF are also computed at the same time. Similarities are computed following Equation 6.2. Now we describe the supervised learning process that we use to compute the weight parameters.

The inputs to the process include: the $m \times (n + c)$ rating matrix $\mathbf{R}$, a precomputed $(n + c) \times (n + c)$ similarity matrix $\mathbf{S}$ based on Equation 6.2 (there

is a $c \times c$ blank sub-matrix because similarity between context items are not required). The output will be an $m \times c$ weight matrix $\mathbf{W}$.

We explain the learning process for a specific user u. This will generate one row of matrix $\mathbf{W}$. The complete matrix is obtained after applying the learning process for all users. We define:

$$\mathbf{w}_u = \begin{bmatrix} w_{u,C_1} & w_{u,C_2} & \dots & w_{u,C_c} \end{bmatrix}^\intercal \tag{6.4}$$

$$\mathbf{s}_i = \begin{bmatrix} s_{i,C_1} & s_{i,C_2} & \dots & s_{i,C_c} \end{bmatrix}^\intercal \tag{6.5}$$

$$\mathbf{R}_u = \begin{bmatrix} R_{u,C_1} & R_{u,C_2} & \dots & R_{u,C_c} \end{bmatrix}^\intercal \tag{6.6}$$

Substituting known values with constants:

$$P_{u,i} = \frac{C_{1,u,i} + \mathbf{w}_u{}^\intercal(\mathbf{s}_i \circ \mathbf{R}_u)}{C_{2,u,i} + \mathrm{tr}(|\mathbf{w}_u \mathbf{s}_i{}^\intercal|)} \tag{6.7}$$

where $\mathbf{s}_i \circ \mathbf{R}_u$ is the entrywise product of s_i and R_u, $\mathrm{tr}(\mathbf{A})$ is the trace of a matrix $\mathbf{A}$.

Suppose the number of items rated by user u is n_u. Their indices are from 1 to n_u. Then we can list an array of equations based on Equation 6.7 as a constant in each iteration. Letting $P_{u,i} = R_{u,i}$ for $i \in \{1, 2, \dots, n_u\}$, the set of equations will have the form:

$$\mathbf{A}\mathbf{w}_u = \mathbf{b} \tag{6.8}$$

where $\mathbf{A}$ is a $n_u \times ||\mathbf{C}||$ coefficient matrix, and $\mathbf{b}$ is a coefficient vector of length

Input: rating matrix $\mathbf{R}$, similarity matrix $\mathbf{S}$, user count m, normal item count n, context count c, threashold t
Output: weight matrix $\mathbf{W}$
for $u = 1$ **to** m **do**
Initialize $\mathbf{w}_u^{(0)} = \begin{bmatrix} w_{u,C_1} & w_{u,C_2} & \dots & w_{u,C_c} \end{bmatrix}^\intercal = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^\intercal$;
$l = 1$;
repeat
for each $R_{u,i} \neq 0$, $1 \leq i \leq n$ **do**
Compute $C_{1,u,i} = \sum_{N_j \in \mathbf{N}} s_{i,N_j} * R_{u,N_j}$;
Compute $C_{2,u,i} = \sum_{N_j \in \mathbf{N}} (|s_{i,N}|)$;
Let $P_{u,i} = \dfrac{C_{1,u,i} + \sum_{C_j \in \mathbf{C}} w_{u,C_j}^{(l)} * s_{i,C_j} * R_{u,C_j}}{C_{2,u,i} + \sum_{C_j \in \mathbf{C}} (|w_{u,C_j}^{(l-1)} * s_{i,C_j}|)} = R_{u,i}$;
Formulate this equation as: $\mathbf{A}_{u,i}^\intercal \mathbf{w}_u^{(l)} = b_{u,i}$, where $\mathbf{A}_{u,i}$ is a vector of length c, $b_{u,i}$ is a scalar;
end for
Combine the equations to yield the linear equation $\mathbf{A}\mathbf{w}_u^{(l)} = \mathbf{b}$, where $\mathbf{A} = \begin{bmatrix} \mathbf{A}_{u,1} & \mathbf{A}_{u,2} & \dots & \mathbf{A}_{u,c} \end{bmatrix}^\intercal$, $\mathbf{b} = \begin{bmatrix} b_{u,1} & b_{u,2} & \dots & b_{u,c} \end{bmatrix}^\intercal$;
Solve for $\mathbf{w}_u^{(l)} = \mathbf{A}^+\mathbf{b}$;
$\Delta\mathbf{w}_u = |\mathbf{w}_u^{(l)} - \mathbf{w}_u^{(l-1)}|$;
until $\Delta\mathbf{w}_u < t$
Save $\mathbf{w}_u = \mathbf{w}_u^{(l)}$;
end for
$\mathbf{W} = \begin{bmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \dots & \mathbf{w}_m \end{bmatrix}^\intercal$;
return $\mathbf{W}$;

Figure 6.1: Learning Process for Weight Parameters

$||\mathbf{C}||$. Then, we can use Moore-Penrose pseudo-inverse to get $\mathbf{w}_u$ for the next iteration:

$$\mathbf{w}_u = \mathbf{A}^+\mathbf{b} \tag{6.9}$$

The above mentioned process is formulated in Fig 6.1.

6.3.3 Sparsity Problem

Sparsity problem is well-known in the community of recommendation systems. In our system, we tackle the sparsity problem by leveraging the power of semantic web.

If the rating matrix is not sparse (specifically, the number of ratings given by user u is greater than δ), our modified CF algorithm is applied to give recommendations. Otherwise, out system will work in the expert system mode. The system will first ask the user to input several selecting criteria. The input UI is demonstrated as in Fig 6.2. The UI prompts user to input a searching criteria, represented by a subject, a property and an object/value. The top spinner (drop-down list) contains all ontology classes. They are sorted according to the class hierarchy to ensure easy access. The second spinner specifies a restriction on ontology properties. Initially this spinner contains all possible properties, but as users select some specific class in the first step, some properties that cannot be applied to the specified class are filtered out. Depending on whether the property selected is an *ObjectProperty* or *DatatypeProperty*, the user will be prompted to input either the third spinner or the text field. Inside the text field, users can input either a number, or an expression (for example "> 1"). After the selection criteria is determined, a semantic language query is fired to retrieve all relevant items matching those ontology classes, and the answers are formatted and displayed to users. The number of ratings received for each item in the answer set is used to determine the order by which these answers are displayed to users.

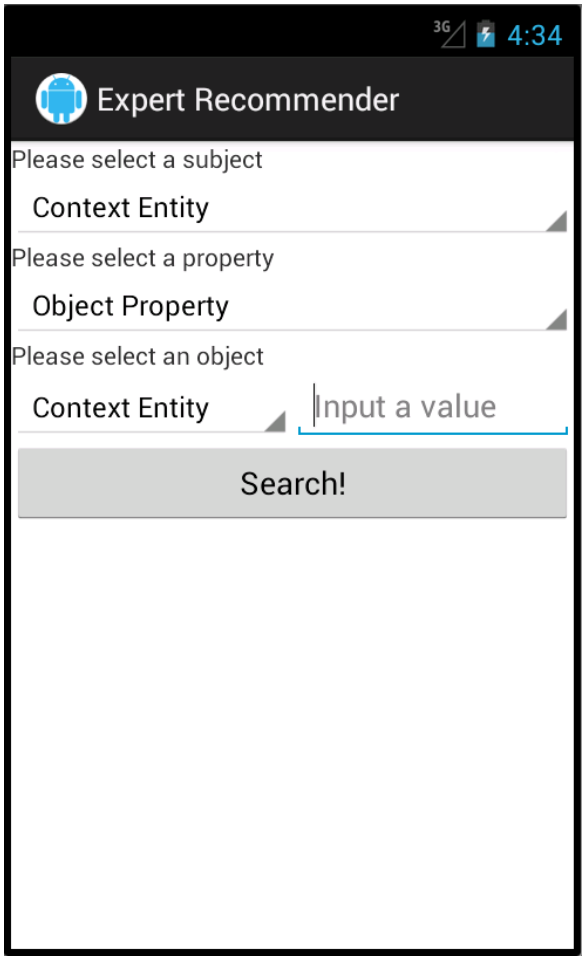

Figure 6.2: When the rating matrix is sparse, a search dialog is present

6.4 Evaluation

This section evaluates our system in two perspectives. Firstly, we examine the effectiveness of the recommendation algorithm as well as the learning process through quantitative metrics. Secondly, we survey users of the prototype system to receive qualitative responses.

6.4.1 Effectiveness of the Algorithm

The differences between CCF and other multi-dimensional (MD) context-aware CF algorithms are: 1. CCF incorporates context information into consideration. 2. A weighting scheme is employed to translate the impact of context information into prediction values. 3. An iterative learning process is devised to produce the weights.

In order to compare the performance of CCF with MD, a dataset with context information appended to each rating is required. However, traditional recommendation system benchmarks do not consider the context of the rating. We have to collect the information by ourselves.

We built a spreadsheet to collect user ratings on movies. By referring to the data collection procedure as described in [104] and [105], each rating is appended with 4 context attributes: Gender, Time (weekday or weekend), Location (at home or at cinema), and Companion (alone, with friends, with lover, or with family). All of the context information can be input with the possible choice of "don't remember". Finally, a dataset with 52 users, 38 movies and 945 ratings is constructed.

The evaluation of our system is carried out as follows: The ratings given by 52 users are split into two groups. The one with 45 users is used as training set and the other with 7 users is test set. This split is done 15 times. The metrics chosen are Mean Absolute Error (MAE), Precision and Recall. MAE is defined

as:

$$\text{MAE} = \frac{\sum_{i \in \mathbf{T}} |R_i - P_i|}{||\mathbf{T}||} \tag{6.10}$$

where $\mathbf{T}$ is the test set, R_i is the actual rating, and P_i is the predicted rating.

Precision and recall are based on the assumption that a rating higher or equal to 4 is considered "good". Precision is defined as the portion of truly "good" ratings among the ones that are predicted to be good. Recall is defined as the portion of correctly predicted "good" ratings among all the truly "good" items. Specifically,

$$\text{Precision} = \frac{tp}{tp + fp} \tag{6.11}$$

$$\text{Recall} = \frac{tp}{tp + fn} \tag{6.12}$$

$$\text{F-measure} = \frac{1}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}} \tag{6.13}$$

In this definition, tp means *true positive*, fp means *false positive*, and fn means *false negative*. F-measure is the harmonic mean of precision and recall.

Fig 6.3 shows the comparative MAEs CCF and MD under all 15 experiments. The average MAE is 0.55 for CCF and it is 0.63 for MD. Fig 6.4 shows the precision/recall measure of both approaches. The precision/recall for CCF is averaged to 0.625/0.401, while for MD it is 0.575/0.415. The F-measures are 0.53 and 0.48 for CCF and MD respectively. Though there are exceptions when

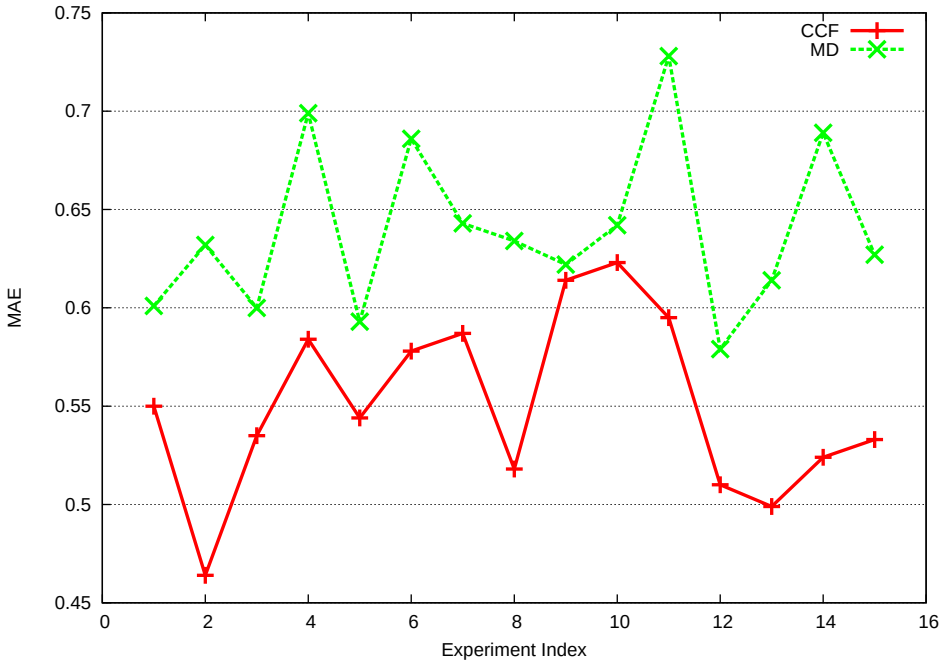

Figure 6.3: MAE measure with 15 splits of the dataset

CCF performs worse than MD, the overall performance of CCF is better, with smaller MAE and higher F-measure.

To sum up, by utilizing our approach, context information can be better utilized and thus better recommendation performance is observed.

6.4.2 User Survey

So far we have justified the effectiveness of our recommendation algorithm, now we want to examine how the system as a whole can help users. A prototype system is built on Android and a server is set up to respond to queries from the clients. The knowledge base of the system is based on the test bench proposed in Chapter 4. Currently we have implemented only one feature of the whole system, i.e. context-aware music album recommendation. For our system, we added

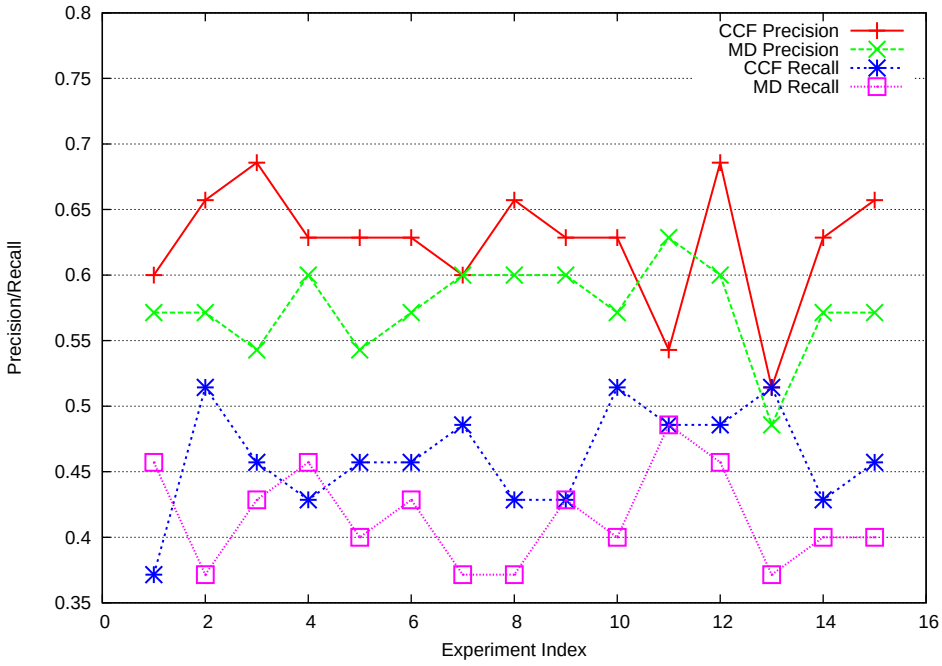

Figure 6.4: Precision and Recall Performance

a thorough hierarchy of modern music genres and albums into the ontology. Specifically, DBPedia[1] is linked to the ontology to provide the professional music taxonomy. The information of a total of 1545 albums released in the years 2010, 2011 and 2012 are extracted from Wikipedia lists.

The system works as follows: At the initial setup, the application will require users to input several ratings for the albums she had listened to. This can help the system to recommend with better precision. However, this step can be skipped when the user prefers to try out the application first. Then the system will recommend several music albums to the user following the algorithm we have proposed. This can be done either through a collaborative filtering process or through the expert system approach. The expert system approach requires the

[1]http://wiki.dbpedia.org/About

user to input several choosing criteria. If no criterion is input, the most popularly recommended albums are recommended to the user. When the recommended item is decided, a Google link and a YouTube link are given to redirect users to listening pages. After the user has finished listening to the music, the users can input her rating for the album in the application. This rating is taken down together with the current context of the user, to improve future recommendation precision.

10 users are invited to try out this application after a prototype has been developed. Some of the user feedbacks are quoted below:

"Interesting app. Hope to see it in one piece and will definitely try again then."

"It gets me introduced to some rock music that I have never tried. But it turns out to be quite good."

"The idea is good. But the application seems to be too dull. More background information could be given when recommending so that we get more than just an album name."

Overall, the feedback is positive. Users are delighted to use the application and many suggestions on improving it are given.

6.5 Chapter Summary

In this chapter, we have proposed a novel system to realize context-aware mobile commerce. Two important aspects of the system are analyzed. The first is how to capture and represent context information. The second is how to make use of

the context information in a recommendation algorithm.

Context information, after being captured by a sensor or a crawler, is represented as a triple in the knowledge base. This triple is then quantified into a scale from 1 to 5, and it is plugged in the rating matrix. Context information is then used in the context-aware collaborative filtering algorithm to tune recommendations. Unlike the traditional solution for context-aware collaborative filtering algorithms, our approach managed to represent context information within 2-D space. With a specially designed weighting scheme, the context information can be utilized in the calculation of similarities between items. This change rid us from setting an arbitrary numerical threshold or a cut-off qualitative context, and we can benefit from the quantitative effect of context information. Together with the weighting scheme, an iterative learning procedure is proposed to learn the weights from training sets.

The algorithm is tested in a movie recommendation scenario. Experiment results show our approach can decrease MAE and produce higher precision and recall. A prototype system on the domain of music recommendation is constructed and multiple users are invited to try and comment on it. The feedback from users shows the system is promising and it gives them positive mobile commerce experiences.

Chapter 7

Conclusion

7.1 Thesis Contribution

In this thesis, we have presented various aspects of context-aware mobile applications.

Firstly, in order to facilitate researches on context-aware mobile applications, we have constructed an ontology-based test bench on the domain of context-awareness. This test bench terminates the time when no ontology-based benchmark is available on the domain of context-awareness and makes developers' lives easier. With this test bench, the research cycle in this area can be greatly shortened. Without putting the effort to build a real system, one can test out many research ideas on this test bench. Through an extensive survey of state-of-the-art mobile applications, the domain of knowledge in this area is modelled as an ontology. A large amount of synthetic concrete information composed using this vocabulary is populated, together with several sample applications, query

sequences and knowledge duplicates.

After that, another important problem while deploying ontology-based context-aware system is studied. With the introduction of our distributed computing scheme, client-server queries can now be classified into different groups and the queries can be handled by many lightweight sub-servers. In designing the distributed computing scheme, a series of problems are tackled. These include 1. how to extract a sub-database from the whole one to answer one category of queries; 2. how to control the size of the sub-database so that we won't end up with a sub-database equal to or even bigger than the whole database; and 3. how to keep the sub-databases synchronized when database updates are required. When we see the thousandfold improvement in processing speed, the overhead of the algorithm is measured to be tens of times larger storage. While this result is by itself acceptable already, we can further reduce the overhead by applying a trade-off, exchanging some overshooting completeness for the benefit of smaller sub-databases. By restricting the maximum number of iterations in filter expansion, the algorithm can be suspended prematurely. Though there is the risk of future data structure changes, this technique will result in a smaller domain of discourse, a smaller sub-database, and lower deployment costs.

The last part of the thesis addresses the problem of introducing context information in mobile recommendation systems. We proposed a novel algorithm that is both able to represent context information in the framework of recommendation, and it is able to make use of the information to improve recommendation precision. A special feature of the algorithm is that we have managed to make

use of the context information in 2-D space. Thus, we have avoided the direct projection of context-enabled 3-D rating matrix to 2-D, thus avoided the information loss. Experimental results show that our context-aware recommendation algorithm can reduce MAE while improving precision and recall. A prototype system on the domain of music recommendation is constructed and multiple users are invited to try and comment on it. The feedback from users shows the system is promising and it gives them positive mobile commerce experiences.

Let's revise the challenges we have tabulated in the first chapter. On the domain of ontology-based context-aware systems, we are facing heterogeneous context information types, slow reasoning speed for real-time requirements, and convoluted ontology structure. For these challenges, we used the OWL DL language to represent various context types, proposed the distributed computing scheme to boost the query processing speed, and resolved the convoluted ontology structure by carefully analyzing the semantics behind. On the domain of context-aware recommendation systems, we have the following challenges: 1. How to incorporate context information in the collaborative filtering is unknown. 2. Existing context-aware recommendation algorithms require hand-picked thresholds to work. 3. The well-known sparsity problem also exists in context-aware recommendation algorithms. For these challenges, we introduce context information as special items and unfold duplicated users to incorporate context information. A weighting scheme is applied to remove the requirement of hand-picked rules. Lastly, we use an ontology-based expert system to replace the recommendation algorithm when the data is sparse.

This thesis as a whole solves some technical problems in context-aware mobile applications. It also intends to promote the usage of such applications. We hope our work can inspire other researchers as well as application developers to contribute to the course of pervasive computing. Let's expect our life styles to be totally different in 10 to 20 years from now thanks to the pervasive computing vision.

7.2 Future Work

This thesis solved various problems in the domain of context-aware mobile applications. However, there are still many issues left ot be addressed. Future research directions can be:

1. Extend the partitioning algorithm introduced in Chapter 5 to incorporate changes in OWL 2. OWL 2 is standardized at the end of 2012, so it's not considered when that part of work is completed. OWL 2 introduced some more semantics and restrictions as compared to OWL, thus the algorithm should be updated as well.

2. Continue building the music recommendation application introduced in Chapter 6. Currently we have finished validating the context-aware recommendation algorithm as well as a user survey for the expert system. However, one can continue to develop this application, introducing the context-aware recommendation algorithm in the system. When all the features are completed, I believe the application would be a popular one

among music lovers.

3. One new research direction is to devise a new temporal context model. Current context models are not very efficient when expressing history contexts. Most often the history contexts are directly replaced with new ones. But it is reasonable to assume the history of contexts can have an impact on future contexts. Therefore, discarding those information would be a waste of resource. This temporal context model should provide means to store as well as reasoning over history context information.

4. With the proliferation of context-aware mobile applications in the future, the generated data set would be growing at a very fast speed. Another research direction is to apply data mining techniques in this ever growing data set to derive higher-level contexts. These higher-level contexts can include users' hidden preferences, or perhaps they can be used to detect life style patterns so even psychological recommendations can be given.

Bibliography

[1] M. Weiser. "The Computer for the 21st Century". In: *Scientific American* 265.3 (Sept. 1991): *Special Issue on Communications, Computers, and Networks*, pp. 94–104.

[2] R. Want et al. "The Active Badge Location System". In: *ACM Transactions on Information Systems (TOIS)* 10.1 (1992), p. 102.

[3] R. Want et al. "The PARCTAB Ubiquitous Computing Experiment". In: *Kluwer International Series in Engineering and Computer Science* (1996), pp. 45–97.

[4] I. Li, A.K. Dey, and J. Forlizzi. "Using context to reveal factors that affect physical activity". In: *ACM Transactions on Computer-Human Interaction* 19.1 (May 2012), 7:1–7:21.

[5] A.K. Dey and G.D. Abowd. "Towards A Better Understanding of Context and Context-awareness". In: *CHI 2000 Workshop on the What, Who, Where, When, and How of Context-Awareness*. ACM Press, 2000, pp. 304–307.

[6] A.K. Dey. “Understanding and Using Context”. In: *Personal and Ubiquitous Computing* 5.1 (2001), pp. 4–7.

[7] J. Hong, E. Suh, and S.J. Kim. “Context-aware Systems: A Literature Review and Classification”. In: *Expert Systems with Applications* 36.4 (2009), pp. 8509–8522.

[8] E. Sunagawa et al. “An Environment for Distributed Ontology Development Based on Dependency Management”. In: *Proceedings of the Second International Semantic Web Conference (ISWC ’03)*. Springer-Verlag. 2003, p. 453.

[9] M. Baldauf, S. Dustdar, and F. Rosenberg. “A Survey on Context-aware Systems”. In: *International Journal of Ad Hoc and Ubiquitous Computing* 2.4 (2007), pp. 263–277.

[10] T. Strang and C. Linnhoff-Popien. “A Context Modeling Survey”. In: *Workshop on Advanced Context Modelling, Reasoning and Management as part of UbiComp*. 2004.

[11] D. Salber, A.K. Dey, and G.D. Abowd. “The Context Toolkit: Aiding the Development of Context-enabled Applications”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM. 1999, p. 441.

[12] A.K. Dey and G.D. Abowd. “The Context Toolkit: Aiding the Development of Context-aware Applications”. In: *Workshop on Software Engineering for wearable and pervasive computing*. 2000, pp. 431–441.

[13] A.K. Dey, G.D. Abowd, and D. Salber. "A Conceptual Framework and A Toolkit for Supporting the Rapid Prototyping of Context-aware Applications". In: *Human-Computer Interaction* 16.2 (2001), pp. 97–166.

[14] G. Klyne et al. *Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0*. World Wide Web Consortium. Jan. 2004. URL: http://www.w3.org/TR/CCPP-struct-vocab/.

[15] A. Held, S. Buchholz, and A. Schill. "Modeling of Context Information for Pervasive Computing Applications". In: *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI)*. 2002.

[16] Ekaterina Chtcherbina and Marquart Franz. "Peer-to-peer coordination framework (p2pc): Enabler of mobile ad-hoc networking for medicine, business, and entertainment". In: *Proceedings of International Conference on Advances in Infrastructure for Electronic Business, Education, Science, Medicine, and Mobile Technologies on the Internet*. LAquila, Italy, 2003, pp. 883–891.

[17] K. Henricksen, J. Indulska, and A. Rakotonirainy. "Generating context management infrastructure from high-level context models". In: *Industrial Track Proceedings of the 4th International Conference on Mobile Data Management (MDM2003)*. 2003.

[18] K. Henricksen and J. Indulska. "Developing context-aware pervasive computing applications: Models and approach". In: *Pervasive and Mobile Computing* 2.1 (2006), pp. 37–64.

[19] T. Hofer et al. “Context-awareness on Mobile Devices - the Hydrogen Approach”. In: *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*. 2002, pp. 292–302.

[20] U. Varshney, R.J. Vetter, and R. Kalakota. “Mobile commerce: a new frontier”. In: *Computer* 33.10 (2000), pp. 32–38.

[21] E.W.T. Ngai and A. Gunasekaran. “A review for mobile commerce research and applications”. In: *Decision Support Systems* 43.1 (2007), pp. 3–15.

[22] Y.F. Chang, C.S. Chen, and H. Zhou. “Smart phone for mobile commerce”. In: *Computer Standards & Interfaces* 31.4 (2009), pp. 740–747.

[23] F. Baader et al. *The Description Logic Handbook*. 2nd ed. New York, NY, USA: Cambridge University Press, 2007.

[24] X.H. Wang et al. “Ontology Based Context Modeling and Reasoning using OWL”. In: *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*. IEEE Computer Society. 2004, p. 18.

[25] R.M. Smullyan. *First-order logic*. Dover Publications, 1995.

[26] M. Krötzsch, F. Simančík, and I. Horrocks. *A Description Logic Primer*. Tech. rep. abs/1201.4089. CoRR, 2012.

[27] I. Horrocks, P.F. Patel-Schneider, and F. Van Harmelen. “From SHIQ and RDF to OWL: The Making of a Web Ontology Language”. In: *Web se-*

mantics: science, services and agents on the World Wide Web 1.1 (2003), pp. 7–26.

[28] D. Beckett and B. McBride. *RDF/XML Syntax Specification (Revised)*. World Wide Web Consortium. Feb. 2004. URL: http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/.

[29] G. Klyne, J.J. Carroll, and B. McBride. *Resource Description Framework (RDF): Concepts and Abstract Syntax*. World Wide Web Consortium. Feb. 2004. URL: http://www.w3.org/TR/rdf-concepts/.

[30] D. Brickley, R.V. Guha, and B. McBride. *RDF Vocabulary Description Language 1.0: RDF Schema*. World Wide Web Consortium. Feb. 2004. URL: http://www.w3.org/TR/2004/REC-rdf-schema-20040210/#ch_domain.

[31] F. Baader et al. *The Description Logic Handbook*. 2nd ed. New York, NY, USA: Cambridge University Press, 2007. Chap. 3, pp. 105–148.

[32] Ian Dickinson. *The Jena Ontology API*. Feb. 2009. URL: http://jena.sourceforge.net/ontology/index.html.

[33] V. Haarslev and R. Möller. "Racer: An OWL reasoning agent for the semantic web". In: *Proceedings of the International Workshop on Applications, Products and Services of Web-based Support Systems*. 2003, pp. 91–95.

[34] I.R. Horrocks. “Using an expressive description logic: FaCT or fiction?” In: *Proceedings of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning*. 1998, pp. 636–647.

[35] D. Tsarkov and I. Horrocks. “FaCT++ description logic reasoner: System description”. In: *Automated Reasoning* (2006), pp. 292–297.

[36] J. Zhou et al. “Minerva: A Scalable OWL Ontology Storage and Inference System”. In: *The Semantic Web - ASWC 2006*. Ed. by Riichiro Mizoguchi, Zhongzhi Shi, and Fausto Giunchiglia. Vol. 4185. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2006, pp. 429–443.

[37] R. Shearer, B. Motik, and I. Horrocks. “Hermit: A highly-efficient owl reasoner”. In: *Proceedings of the 5th International Workshop on OWL: Experiences and Directions (OWLED 2008)*. 2008.

[38] C. Golbreich and E.K. Wallace. *OWL 2 Web Ontology Language New Features and Rationale*. Oct. 2009. URL: http://www.w3.org/TR/owl2-new-features/.

[39] E. Prud’hommeaux and A. Seaborne. *SPARQL Query Language for RDF*. World Wide Web Consortium. Jan. 2008. URL: http://www.w3.org/TR/rdf-sparql-query/.

[40] A. Seaborne et al. *SPARQL Update*. July 2008. URL: http://www.w3.org/Submission/SPARQL-Update.

[41] Jerome F. DiMarzio. *Android: A Programmer’s Guide*. McGraw-Hill, 2008.

[42] Zigurd Mednieks et al. *Programming Android: Java Programming for the New Generation of Mobile Devices*. O'Reilly, 2012.

[43] Joe Conway and Aaron Hillegass. *iOS programming: the big nerd ranch guide*. Addison-Wesley Professional, 2011.

[44] Alasdair Allan. *Basic Sensors in iOS: Programming the Accelerometer, Gyroscope, and More*. O'Reilly, 2011.

[45] Microsoft. *Windows Phone Dev Center*. 2013. URL: http://developer.windowsphone.com/en-us.

[46] Tor-Morten Grønli, Jarle Hansen, and Gheorghita Ghinea. "Android vs Windows Mobile vs Java ME: A Comparative Study of Mobile Development Environments". In: *Proceedings of the 3rd International Conference on PErvasive Technologies Related to Assistive Environments (PETRA '10)*. ACM, 2010, 45:1–45:8.

[47] Juan Ye et al. "Ontology-based models in pervasive computing systems". In: *The Knowledge Engineering Review* 22.4 (2007), pp. 315–347.

[48] J. Indulska et al. "Experiences in using CC/PP in context-aware systems". In: *Proceedings of the 4th International Conference on Mobile Data Management*. Vol. 2574. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2003, pp. 247–261.

[49] S. Buchholz, T. Hamann, and G. Hübsch. "Comprehensive Structured Context Profiles (CSCP): Design and Experiences". In: *Proceedings of*

the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops. IEEE Computer Society. 2004, p. 43.

[50] J. Bacon, J. Bates, and D. Halls. "Location-oriented multimedia". In: *IEEE Personal Communications* 4.5 (1997), pp. 48–57.

[51] T. Gu, H.K. Pung, and D.Q. Zhang. "A service-oriented middleware for building context-aware services". In: *Journal of Network and computer applications* 28.1 (2005), pp. 1–18.

[52] T. Gu and H.K. Pung. "A middleware for building context-aware mobile services". In: *Proceedings of the 59th IEEE Vehicular Technology Conference*. Vol. 5. 2005, pp. 2656–2660.

[53] H. Chen, T. Finin, and A. Joshi. "An Ontology for Context-aware Pervasive Computing Environments". In: *The Knowledge Engineering Review* 18.3 (2004), pp. 197–207.

[54] J.I. Hong and J.A. Landay. "An Infrastructure Approach to Context-aware Computing". In: *Human-Computer Interaction* 16.2 (2001), pp. 287–303.

[55] P. Fahy and S. Clarke. "CASS-Middleware for Mobile Context-aware Applications". In: *Workshop on Context Awareness, MobiSys*. 2004, pp. 304–308.

[56] M. Román et al. "A middleware infrastructure to enable active spaces". In: *IEEE Pervasive Computing* 1.4 (2002), pp. 74–83.

[57] E. Aitenbichler, J. Kangasharju, and M. Mühlhäuser. “MundoCore: A light-weight infrastructure for pervasive computing”. In: *Pervasive and Mobile Computing* 3.4 (2007), pp. 332–361.

[58] C. Lin et al. “On context-aware distributed event dissemination”. In: *Personal and Ubiquitous Computing* 15.3 (Mar. 2011), pp. 305–314.

[59] G. Chen and D. Kotz. “Solar: An Open Platform for Context-aware Mobile Applications”. In: *Proceedings of the First International Conference on Pervasive Computing (Short paper)*. 2002, pp. 41–47.

[60] Dejene Ejigu, Marian Scuturici, and Lionel Brunie. “An Ontology-Based Approach to Context Modeling and Reasoning in Pervasive Computing”. In: *Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops*. Mar. 2007, pp. 14–19.

[61] P. Korpipää et al. “Managing context information in mobile devices”. In: *IEEE Pervasive Computing* 2.3 (2003), pp. 42–51.

[62] P. Korpipää and J. Mäntyjärvi. “An Ontology for Mobile Device Sensor-Based Context Awareness”. In: *Modeling and using context: 4th International and Interdisciplinary Conference, CONTEXT 2003: Proceedings*. LNAI 2680. Springer-Verlag. June 2003, pp. 451–458.

[63] P. Korpipää et al. “Bayesian approach to sensor-based context awareness”. In: *Personal and Ubiquitous Computing* 7.2 (2003), pp. 113–124.

[64] M. Hatala, R. Wakkary, and L. Kalantari. “Rules and ontologies in support of real-time ubiquitous application”. In: *Web Semantics: Science, Services and Agents on the World Wide Web* 3.1 (2005), pp. 5–22.

[65] M. Strobbe et al. “Novel Applications Integrate Location and Context Information”. In: *IEEE Pervasive Computing* 11.2 (Feb. 2012), pp. 64 –73.

[66] G.D. Abowd et al. “Cyberguide: A Mobile Context-aware Tour Guide”. In: *Wireless Networks* 3.5 (1997), pp. 421–433.

[67] K. Cheverst et al. “Developing a Context-aware Electronic Tourist Guide: Some Issues and Experiences”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM. 2000, p. 24.

[68] P.J. Brown. “Triggering information by context”. In: *Personal and Ubiquitous Computing* 2.1 (1998), pp. 18–27.

[69] F. Calabrese et al. “Estimating Origin-Destination Flows Using Mobile Phone Location Data”. In: *IEEE Pervasive Computing* 10.4 (Oct. 2011), pp. 36–44.

[70] M.C. Chiu et al. “Playful bottle: a mobile social persuasion system to motivate healthy water intake”. In: *Proceedings of the 11th international conference on Ubiquitous computing*. ACM. 2009, pp. 185–194.

[71] B. Ferris, K. Watkins, and A. Borning. “Location-aware tools for improving Public transit Usability”. In: *IEEE Pervasive Computing* 9.1 (2010), pp. 13–19.

[72] S. Ou, K. Yang, and J. Zhang. “An effective offloading middleware for pervasive services on mobile devices”. In: *Pervasive and Mobile Computing* 3.4 (2007), pp. 362–385.

[73] B. Guo, D. Zhang, and M. Imai. “Toward a cooperative programming framework for context-aware applications”. In: *Personal and Ubiquitous Computing* 15.3 (Mar. 2011), pp. 221–233.

[74] J. Zhu et al. “A context realization framework for ubiquitous applications with runtime support”. In: *IEEE Communications Magazine* 49.9 (Sept. 2011), pp. 132–141.

[75] F. Paganelli and D. Giuli. “An Ontology-Based System for Context-Aware and Configurable Services to Support Home-Based Continuous Care”. In: *IEEE Transactions on Information Technology in Biomedicine* 15.2 (Mar. 2011), pp. 324–333.

[76] A.T. Campbell et al. “The Rise of People-centric Sensing”. In: *IEEE Internet Computing* (2008), pp. 12–21.

[77] B. Priyantha, D. Lymberopoulos, and J. Liu. “LittleRock: Enabling Energy-Efficient Continuous Sensing on Mobile Phones”. In: *IEEE Pervasive Computing* 10.2 (2011), pp. 12–15.

[78] M. d’Aquin et al. “Toward a New Generation of Semantic Web Applications”. In: *IEEE Intelligent Systems* 23.3 (May 2008), pp. 20–28.

[79] N. Choi, I. Song, and H. Han. “A survey on ontology mapping”. In: *SIGMOD Rec.* 35.3 (Sept. 2006), pp. 34–41.

[80] S. Amrouch and S. Mostefai. “Survey on the literature of ontology mapping, alignment and merging”. In: *Proceedings of International Conference on Information Technology and e-Services (ICITeS)*. Mar. 2012, pp. 1–5.

[81] M. Bhatt et al. “MOVE: A Distributed Framework for Materialized Ontology View Extraction”. In: *Algorithmica* 45 (3 2006), pp. 457–481.

[82] U. Sumita and J. Yoshii. “Enhancement of E-commerce via mobile accesses to the Internet”. In: *Electronic commerce research and applications* 9.3 (2010), pp. 217–227.

[83] B. Sarwar et al. “Analysis of recommendation algorithms for e-commerce”. In: *Proceedings of the 2nd ACM conference on Electronic commerce*. ACM. 2000, pp. 158–167.

[84] R. Agrawal and R. Srikant. “Fast algorithms for mining association rules”. In: *Proceedings of the 20th International Conference of Very Large Data Bases (VLDB)*. Vol. 1215. 1994, pp. 487–499.

[85] R. Agrawal et al. “Fast discovery of association rules”. In: *Advances in knowledge discovery and data mining* 12 (1996), pp. 307–328.

[86] D. Goldberg et al. “Using collaborative filtering to weave an information tapestry”. In: *Communications of the ACM* 35.12 (1992), pp. 61–70.

[87] J.A. Konstan et al. “GroupLens: Applying collaborative filtering to Usenet news”. In: *Communications of the ACM* 40.3 (1997), pp. 77–87.

[88] W. Hill et al. “Recommending and evaluating choices in a virtual community of use”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM Press. 1995, pp. 194–201.

[89] U. Shardanand and P. Maes. “Social information filtering: algorithms for automating word of mouth”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM Press. 1995, pp. 210–217.

[90] B. Sarwar et al. “Item-based collaborative filtering recommendation algorithms”. In: *Proceedings of the 10th international conference on World Wide Web*. Hong Kong: ACM, 2001, pp. 285–295.

[91] D. Rosaci and G. Sarné. “A multi-agent recommender system for supporting device adaptivity in e-Commerce”. In: *Journal of Intelligent Information Systems* 38 (2 2012), pp. 393–418.

[92] Y. Zhang, Y. Zheng, and J. Ni. “Context-Aware Commodity Recommendation Information Service in E-commerce”. In: *Proceedings of the Sixth International Conference on Internet Computing for Science and Engineering (ICICSE)*. Apr. 2012, pp. 20–25.

[93] A. Chen. “Context-Aware Collaborative Filtering System: Predicting the User’s Preference in the Ubiquitous Computing Environment”. In: *Proceedings of the first international workshop on Location- and context-awareness (LoCA)*. Springer-Verlag. May 2005, p. 244.

[94] P. Lukowicz and S. Intille. “Experimental Methodology in Pervasive Computing”. In: *IEEE Pervasive Computing* 10.2 (2011), pp. 94–96.

[95] Y. Guo, Z. Pan, and J. Heflin. "LUBM: A Benchmark for OWL Knowledge Base Systems". In: *Web Semantics: Science, Services and Agents on the World Wide Web* 3.2 (2005), pp. 158–182.

[96] L. Ma et al. "Towards a complete OWL ontology benchmark". In: *The Semantic Web: Research and Applications*. Vol. 4011. LNCS. Springer, June 2006, pp. 125–139.

[97] Christian B. and Andreas S. "The Berlin SPARQL Benchmark". In: *International Journal on Semantic Web and Information Systems* 5.2 (2009), pp. 1–24.

[98] M.K. Smith, C. Welty, and D.L. McGuinness. *OWL Web Ontology Language Guide*. World Wide Web Consortium. Feb. 2004. URL: http://www.w3.org/TR/2004/REC-owl-guide-20040210/.

[99] P. Hayes and B. McBride. *RDF Semantics*. World Wide Web Consortium. Feb. 2004. URL: http://www.w3.org/TR/2004/REC-rdf-mt-20040210/.

[100] P. F. Patel-Schneider and I. Horrocks. *OWL Web Ontology Language Semantics and Abstract Syntax Section 2. Abstract Syntax*. World Wide Web Consortium. Feb. 2004. URL: http://www.w3.org/TR/2004/REC-owl-semantics-20040210/syntax.html.

[101] B.N. Grosof et al. "Description logic programs: Combining logic programs with description logic". In: *Proceedings of the 12th international conference on World Wide Web*. Budapest, Hungary, 2003, pp. 48–57.

[102] Hewlett-Packard Jena team. *Jena - A Semantic Web Framework for Java*. Aug. 2010. URL: http://jena.sourceforge.net/index.html.

[103] E. Sirin et al. "Pellet: A Practical OWL-DL Reasoner". In: *Web Semantics: Science, Services and Agents on the World Wide Web* 5.2 (2007), pp. 51–53.

[104] G. Adomavicius et al. "Incorporating contextual information in recommender systems using a multidimensional approach". In: *ACM Transactions of Information Systems* 23.1 (Jan. 2005), pp. 103–145.

[105] Z. Huang, X. Lu, and H. Duan. "Context-aware recommendation using rough set model and collaborative filtering". In: *Artificial Intelligence Review* 35 (1 2011), pp. 85–99.

Appendix

Table 7.1: Results of Mobile Application Survey (partial)

App Name	**Use Cases**
Category: Book and References	
Google Sky Map	GPS reading. Accelerometer reading.
Bible	Bible contents online.
Dictionary.com	Online content of dictionary and thesaurus, pronunciation, spelling suggestion, example sentence, etymology, daily content, voice-to-text, favorite word list.
Moon Phase Pro	Show moon phases, crescent angle, rise/set times. Calendar show, live wallpaper, widgets.
Aldiko Book Reader Premium	Read and download ebooks. Import your own ePub and pdf files.

Continued on next page

Table 7.1 – continued from previous page

App Name	Use Cases
Category: Business	
Documents To Go 3.0 Main App	View, create and modify Microsoft word, excel and powerpoint files. Rich formatting in word, many functions support in excel, view and rehearse powerpoint, google docs support, desktop sync, attachment, password-protected files
Exchange for Android 2.x	Touchdown syncs email, contacts, calendar, tasks, notes and SMS.
PrinterShare Mobile Print	Print service to nearby direct printing via wifi or bluetooth. Print to nearby by PC shared printer. Print to remote PC shared printers.
CamCard - Business Card Reader	Capture camera image of business cards. Image enhancement. Save into contacts(phone, gmail, exchange). QR code generation and recognition. Email signature recognition. Linkedin invitation.
Category: Communication	
Torque	A car performance diagnosis tool. It measures torque, bhp, temperature, rpm. Fusion with Google earth. See your car's status in real time.

Continued on next page

Table 7.1 – continued from previous page

App Name	Use Cases
Backup to Gmail	Sync SMS, MMS, and call logs to gmail account.
WebSharing File/Media Sync	Share files between phone and computer. Set up a temporary http server on mobile phone, and access it using a generated URL on computer to do syncing.
WhatsApp Messenger	Real-time messenger, push notification, server storage, group chat, exchange contact.
Category: Finance	
Pageonce Pro - Money & Bills	Manage your bank accounts, credit cards, bills and investments in one place. Real-time alerts or notifications.
anMoney	Personal finance assist with syncing ability. Calendar. Send event to guests. Import payee from contact. Budgeting.
Chase Mobile	JP Morgan Chase accounts. Nearest branch or atm locator. Talk to service representatives. Deposit checks.
Square	Personal payment terminal. Credit card reader. Accept visa, master, and many more.
Category: Health & Fitness	

Continued on next page

Table 7.1 – continued from previous page

App Name	Use Cases
Endomondo Sports Tracker Pro	GPS tracking of time, distance, speed, calories. Audio feekback every mile or km. Workout route map. Friends list. Beat friends. Share on facebook. Personal history. Time goal.
CardioTrainer Pro	Weight loss trainer, measure heart beats, track route, voice output and music, pro training of 20 levels.
Calorie Counter Pro	416,000+ foods database, search food, favorite and typical serving, recipe, activities(exercise), weight-loss plan.
Baby ESP	Track baby's activities, including nap, sleep, breast feeding, bottles, diapers, medicines, breast pumping. Compare growth with WHO growth chart. Sync data between devices. Reminder notification. Keep journal. Compare with friends.
Category: Libraries & Demo	
eSpeak for Android	Port of eSpeak engine on android. Text-to-speech.
ES Security Manager (beta)	Protect privacy (password to SMS, dialer, contacts). Scan threats. Find lost phones (lock remotely, get location, SMS, contacts, SIM information back).
Category: Lifestyle	

Continued on next page

Table 7.1 – continued from previous page

App Name	Use Cases
Sleep as an Droid	Use accelerometer to track movement when sleeping. These movings are modelled to match your sleeping pattern, and wakes you up when in light sleep.
Jamie's 20 Minute Meals	A large amount of recipe data that goes with many video illustrations.
Horoscope	Horoscope texts for today, tomorrow, and for current month. Updated everyday.
Zillow Real Estate & Rentals	Estimate of home value and rent. Home for sale/rent information. Near-by homes/apartments.
Category: Media & Video	
iSyncr	Sync iTunes playlist to phones.
Ringdroid	Create ringtones by cutting audio files or record on the fly.
MagicMarker	Touch-paint program for writing and drawing neon-style on black background. Set as background or share through mail or SNS.
DoggCatcher Podcast Player	Feeds and podcasts reader/player.
Category: Medical	
ICE: In Case of Emergency	A list of people to call in emergency. Insurance information. Doctor names and numbers.

Continued on next page

Table 7.1 – continued from previous page

App Name	Use Cases
Medscape	Drug references, drug interaction checker, disease and condition reference and treatment guide, procedure reference, daily medical news, physician/pharmacy/hospital directories.
Mini Nurse - Lite	Medication dosage, IV rate, nursing skill.
iPharmacy: Pill ID & Rx ref	Bar-code reader for drugs. Detailed drug guide. Indication, dosage, contraindication, precautions, adverse reaction, drug interaction, overdosage, and how-supplied of pills and drugs.
Category: Music & Audio	
PowerAMP Music Player	Music player. Equalizer. Download missing album art. Visual themes.
SoundHound	Music recognition, hum a tune to search. Instant lyrics and artist information. Voice search for albums and bands.
Shazam Encore	Use a music clip to identify, buy, watch related video, get lyrics, and share with friends.
Pandora(®) Internet Radio	Personalized radio station. Start with the name of your favorite artist, song or composer, Pandora will create a "station" that plays their music and music of same kind.
Category: News & Magazines	

Continued on next page

Table 7.1 – continued from previous page

App Name	Use Cases
Read It Later Pro	Save web contents for later reading. Distilled contents. Sync reading lists. Offline reading. Save scroll position.
NewsRob Pro	Syncs with Google Reader. Downloads full/partial pages of feeds.
World Newspaper	Video news. Translate page. Offline viewing. Read It Later Integration.
Google Reader	Follow favorite sites, blogs. See friends' sharing. Sync.
Category: Photography	
PicSay Pro	Modify and enhance pictures. Sharpen, red eye, crop and stretch, distort, paint, effects, etc.
Vignette	Add film and camera effects to your photos. Effects, frames, different camera styles, timer, geotagging.
Photaf 3D Panorama Pro	Utilize camera and orientation sensor to stitch 3D panorama pictures. Facebook share.
PhotoFunia	Photo editing tool. Auto detects faces and do pasting to interesting backgrounds.
Category: Productivity	
Root Explorer (File Manager)	File manager, SQLite database viewer, text editor, zip file extractor, execute scripts, remount, permission, bookmar, stream files, apk binary XML viewer.

Continued on next page

Table 7.1 – continued from previous page

App Name	Use Cases
Thinking Space Pro	Create visual thought maps to help organize and plan your activities and ideas.
ColorNote	Notes, TODO list, shopping list. Organize scheduler in calendar. Password protection. Reminder on status bar. Search. Colordict add-on. Share notes via SMS, email, twitter.
Google Goggles	Search by real world pictures. Image recognition. Identify products, famous landmarks, storefronts, artwork, popular images. translate. Extract contact info from business cards.
Category: Shopping	
Mighty Grocery Shopping List	Multiple lists, price, quantity, tax, coupon, voice recognition, favorite, sync, barcode scan, recipe.
Barcode Scanner	Scan barcodes on products then look up prices and reviews. Scan Data Matrix and QR Codes and contact info. Share your contacts, apps, and bookmarks via QR Code.
Key Ring Reward Cards	Save loyalty cards and coupons to your phone.
Category: Social	

Continued on next page

Table 7.1 – continued from previous page

App Name	Use Cases
Tapatalk Forum App	Access vB, phpBB, IPB, SMF forums.
SymbolsKeyboard & TextArt Pro	Send ASCII symbols or text art from the library to friends/forums. Create custom art.
FunForMobile	Share ringtone, wallpaper, joke, photo, video. Chat, talk, play games. Download wallpaper, ringtone, video made by other members.
Category: Sports	
SkyDroid - Golf GPS	Satellite view of every golf course, GPS Distance to every green, water hazards, bunkers, etc. Shot Tracking.
Dynomaster	Drag racing application. Data reply, power calculator, satellite and street view, G-meter.
Soccer Score Pro	Live football score, match stats, news(league and club).
Category: Tools	
Titanium Backup PRO	App freezer, multi backups, batch restore, migrate app data across roms.
App Protector Pro	Privacy protection tool. Lock any application on your phone: SMS, Message, Gmail, Photo, Gallery, Market.

Continued on next page

Table 7.1 – continued from previous page

App Name	Use Cases
Category: Transportation	
Car Locator	Parking timer, locate your car, location history, location favorites.
SpeedView Pro	GPS-based speedometer, speed warning.
Plane Finder	Visualize planes on google maps. Plane info.
Waze	Community GPS navigation. User generated traffic info (inclusive of road information, congestion), route time estimation.
Category: Travel & Local	
FlightTrack	Get real-time flight status and map tracking for airline flights worldwide. Delay history, delay forecast.
BackCountry Navigator PRO	Preload topographic map, GPS waypoints, outdoor.
GPS Status & Toolbox	GPS sensor reading, compass with magnetic and true north, leveling tool. Mark and share your location. Navigate back later.
Category: Weather	
WeatherBug Elite	Get the latest weather conditions, forecasts, radar animation, alerts.
The Weather Channel	Forecast temperature, precipitation, wind, UV index, visibility. Video news, integration with iWitness.