A Thesis

For the Degree of Doctor of Philosophy

Context Prediction based Collaborative IoT Architecture for Smart Environment

Rashid Ahmad

Department of Computer Engineering

Graduate School

Jeju National University

Feb 2016

Dedicated to my parents and family,
for their immense support and
encouragement.

Acknowledgment

First and foremost, I render my humble and sincere thanks to the Almighty Allah for showering his countless blessings upon me. He gave me the strength, courage, and patience during this endeavor.

I am deeply indebted to my supervisor Prof. Do-Hyeun Kim for providing me invaluable mentorship, guidance, and support. He provided an inspiring and competitive environment to polish my skills and research interests. He is not only been an excellent scientific mentor, but is a great human being. I am grateful to my dissertation evaluation committee for their insight comments and valuable suggestions during the dissertation defense. It really helped me in elevating the quality of this dissertation. I am grateful to my former lab mates Chen Shu and Nie Yin for their invaluable support during my early days in Jeju-Korea. I am highly obliged to my current lab mates Safdar Ali, Muhammad Sohail Khan, Fazli Wahid, Jin Wenquan, Lei Hang and Hyeun Bok for their priceless suggestions during this endeavor. They are not only colleagues but good friends and always ready to help me. Also, I appreciate all my Korean friends who gave me a wonderful and memorable company during my stay in Jeju.

I have no words to express my sincere gratitude to my parents, sisters and brothers for their endless support, love and prayers. Without them, this thesis would never have been written. This last word of acknowledgment I have saved for my two cute sons (Hafeez and Afaq) to whom I could not give time during this endeavor.

Rashid Ahmad

2016.02

Table of Contents

Acknowledgment **i**

Table of Contents **ii**

List of Figures **vi**

List of Tables **ix**

Abbreviations **x**

Abstract **1**

1. Introduction **3**

1.1. Background 3

1.1.1. IoT architecture standardization 5

1.1.2. State of the art IoT architectures 6

1.1.3. State of the art IoT application platforms 8

1.2. Problem statement 10

1.3. Contribution 11

2. Related work **13**

3. Collaborative IoT architecture (Service-oriented IoT architecture) **17**

3.1. Application layer 17

3.2. Service layer 19

3.2.1. Common Services Components 22

3.2.1.1. Service Registration (SR) 22

3.2.1.2. Service Discovery (SD) 23

3.2.1.3. Service Profiler (SP) 23

3.2.1.4. Service Matching (SM) 24

3.2.1.5. Service Selection (SS) 26

3.2.1.6. Services Clustering (SCG) 26

3.2.1.7. Services Performance Evaluation (SPE) 28

3.2.1.8. Shared Services Repository (SSR) 29

3.2.1.9. Shared Services Profile (SSP) 30

3.2.1.10. Service Composition (SCN) 30

3.2.2. Application Specific Services Components 36

3.2.2.1. Activities Context Collector (ACC) 36

3.2.2.2. Activity Notifications 37

3.2.2.3. Activities Services Orchestration (ASO) 37

3.2.2.4. Activities Local Repository (ALP) 38

3.2.2.5. Energy Context Collector 39

3.2.2.6. Energy Consumption Notifications 40

3.2.2.7. Energy Services Orchestration (ESO) 40

3.2.2.8. Energy Local Repository 41

3.2.2.9. Entertainment Context Collector 42

3.2.2.10. Context based Movies Recommendation Notifications 42

3.2.2.11. Entertainment Services Orchestration (EtSO) 42

3.2.2.12. Entertainment Local Repository (EtLP) 43

3.3.1. Common Processing Components 46

3.3.1.1. Outlier Removal (OR) 46

3.3.1.2. Handling Missing Values (HMV) 46

3.3.1.3. Normalization 47

3.3.1.4. Data Fusion 47

3.3.1.5. Aggregation 47

3.3.1.6. Feature Extraction 48

3.3.1.7. Pre-Process Selector (PPS) 50

3.3.1.8. Data Smoothening (DS) 50

3.3.2. Application Specific Processing Components 51

3.3.2.1. Activities Specific Processing Components 51

3.3.2.2. Energy Specific Processing Components 57

3.3.2.3. Entertainment Specific Processing Components 62

4. Experiment 1: Collaborative context prediction in smart office 70

4.1. Data set and experimental configuration.....80

4.2. Evaluation mechanism 80

4.3. Result and discussions.....81

5. Experiment 2: Energy prediction and analysis in residential smart buildings 89

5.1. Data set and experimental configuration.....105

5.2. Evaluation mechanism 107

5.3. Results and discussion.....108

5.3.1. Data Pre-processing 108

5.3.2. Neural Network in processing layer 113

5.3.3. Classification And Regression Tree (CART) in processing layer 118

5.3.4. Genetic Programming in processing layer 123

5.3.5. Incremental learning based neural network in processing layer 126

5.3.6. Ensemble Model in processing layer 127

6. Experiment 3: Multimedia content recommendation in smart spaces 130

6.3. Data set and experimental configuration.....149

6.4. Evaluation mechanism 155

6.5. Results and discussion.....157

6.5.1. Multimedia content recommendation results 157

6.5.2. Deep learning based recommendation results 162

7. Conclusion 165

References 169

List of Figures

Figure 3.1: Layered component architecture of the CIoT 18

Figure 3.2: Service layer flow model 21

Figure 3.3: Static service composition architecture 32

Figure 3.4: Atomic service generation mechanism 34

Figure 3.5: Activity rules generation flow diagram 36

Figure 3.6: Rules utilization flow diagram 36

Figure 3.7: Processing layer flow model 45

Figure 4.1: Conceptual component diagram of the activity prediction in smart office 71

Figure 4.2: Illustration of overlapped activities 75

Figure 4.3: Mean of raw accelerometer data 82

Figure 4.4: Raw accelerometer data with activity labels 82

Figure 4.5: Confusion matrix for kNN based activity annotation 83

Figure 4.6: The accuracy graph of PCCP for the single user's environment 84

Figure 4.7: Accuracy of prediction by PCCP in collaborative environment 85

Figure 4.8: Comparison of prediction accuracy between with and without collaboration PCCP [8]. 86

Figure 4.9: Time based comparison of CCP and PCCP 87

Figure 5.1: Component architecture for electricity consumption forecasting. 90

Figure 5.2: Flow model of the electricity consumption forecasting. 93

Figure 5.3 Flow diagram of the neural network based model learning 95

Figure 5.4: Flow diagram for the training of model for single multi-step (hours) ahead forecast 97

Figure 5.5: Conceptual model of self-training 102

Figure 5.6: Flow diagram for incremental learning of neural network for multi-step forecasting 103
Figure 5.7: Ensemble Model 104
Figure 5.8 Basic data statistics 107
Figure 5.9 Effect of moving average on one day consumption (window size =3) 109
Figure 5.10: Loess smoothing effect on a one day consumption 110
Figure 5.11: Lowess smoothing effect on a one day consumption 110
Figure 5.12: Effect of robust loess on one day consumption 111
Figure 5.13 Effect of robust lowess on one day consumption 111
Figure 5.14: Effect of Savitzky golay smoothing on one day consumption 112
Figure 5.15: Effect of Smoothing techniques on a single day consumption 112
Figure 5.16: Error plot for smoothing filters with original data 113
Figure 5.17: Building scale hourly forecasting result of a sample day with CV=2.91 114
Figure 5.18: Floor scale hourly forecasting result of a sample day with CV =14.13 115
Figure 5.19: Apartment scale hourly forecasting result of a sample day with CV =16.73 115
Figure 5.20: Day ahead electricity consumption prediction result by FFNN 117
Figure 5.21: MAPE for day ahead consumption prediction using FFNN 118
Figure 5.22: Cart tree on a one month training data 119
Figure 5.23: Day ahead electricity consumption prediction result by regression tree 122
Figure 5.24: Mean absolute percentage error of day ahead prediction from regression tree 123
Figure 5.25: Day ahead electricity consumption prediction result by GP 123
Figure 5.26 Mean absolute percentage error for day ahead consumption prediction using GP. 124
Figure 5.27: Comparison of MAPE for one day consumption prediction 126
Figure 5.28: Incremental learning results 127
Figure 5.29: Ensemble model(Co-training) results 127

Figure 6.1: Layered component architecture of content recommendation 131
Figure 6.2: Flow model for entertainment content recommendation in smart home 133
Figure 6.4 Conceptual model for intelligent multimedia services 137
Figure 6.5: Flow diagram at the client application 138
Figure 6.6: Flow diagram for local app server 138
Figure 6.7: Flow of context processing at the local app server 143
Figure 6.8: User location service 143
Figure 6.9: Sequence diagram for intelligent multimedia services 145
Figure 6.10: BPM for Smart Service provisioning 147
Figure 6.11: Local App server user interface 150
Figure 6.12: Content registration interface 151
Figure 6.13: Devices gateway interface 151
Figure 6.14: Devices management at device gateway 152
Figure 6.15: Device registration interface 153
Figure 6.16: Smart television emulator 153
Figure 6.17: Client user interface 154
Figure 6.18: Client UI for contents and devices view 155
Figure 6.19: User interface of the global content server 155
Figure 6.20 Accuracy of the system for recommending contents 158
Figure 6.21 Performance comparison 158
Figure 6.22: Precision vs Recall on datasets 159
Figure 6.23: Performance comparison using TP, TN, FP, FN 159
Figure 6.24: Accuracy on the different datasets 160
Figure 6.25: Delay comparison of devices communication 161
Figure 6.26: Device to Server (D2S) time comparison 162

List of Tables

Table 1.1: Comparison of IoT application platforms. 9

Table 4.1 Comparison of with and without collaboration of the context prediction..... 86

Table 5.1 Coefficient of Variation (CV) values at different spatial granularity level..... 114

Table 5.2: Results of NN based forecasting..... 116

Table 5.3: Sample rule set of cart on one month training data..... 120

Table 5.4: MAPE of RTREE 121

Table 5.5: Mean absolute error values for day ahead prediction using GP..... 124

Table 5.6: Comparative results of supervised and semi-supervised approaches 128

Table 6.1 Environmental configuration for the implementation of the multimedia services.... 149

Table 6.2: Parameter setting for experimentation 163

Table 6.3: Recall in the sparse setting for top 300 recommendation by 2 layer model 163

Table 6.4: Impact of layer size on the results..... 163

Abbreviations

ESO	Energy Services Orchestration
ECC	Energy Context Collector
ELR	Energy Local Repository
ASO	Activities Services Orchestration
ACC	Activities Context Collector
ALR	Activities Local Repository
EtSO	Entertainment Services Orchestration
EtCC	Entertainment Context Collector
EtLR	Entertainment Local Repository
SR	Service Registration
SD	Service Discovery
SP	Service Profiler
SM	Service Matching
SS	Service Selection
SCG	Service Clustering
SPE	Service Performance Evaluation
SCN	Service Composition
SSR	Shared Service Repository
SSP	Shared Service Profile
FOIL	First Order Inductive Learner
PRM	Predictive Rule Mining

NN	Neural Network
INN	Incremental Neural Network
RTREE	Regression Tree
GP	Genetic Programming
EM	Ensemble Model
DB	Data bases
ARM	Association Rules Mining
BDML	Bayesian Deep Learning Model
DS	Data Smoothening
DF	Data Fusion
PPS	Pre-Process Selector
FE	Feature Extraction
HMV	Handling Missing Values
OR	Outlier Removal
Accel	Accelerometers
Gyro	Gyroscopes

Abstract

The advancement in sensing and communication technologies keeps adding new dimensions to the world of information and communication technologies (ICTs) i.e. connectivity from anyplace, anytime, to anything. This hyper-connectivity leads to a new technological paradigm called Internet of Things (IoT) which is drastically transforming our commercial, social, and personal sphere. The lack of standard procedure for communication among the constituent heterogeneous parts(devices) of IoT post a number of research challenges like connectivity, data processing, privacy preservation etc. Many research studies have been dedicated to address the communication architecture and data processing problem for these things. These studies resulted in various models which enable connecting things, processing the collected data and proactively responding to the situation. Some of these models are application-specific while others introduce a new terminology i.e. everything as a service (XaaS) into existing Service Oriented Architecture (SOA). However, a purely SOA based service oriented system may be wasting resources on processing unnecessary data. Moreover, the concerns over privacy and data protection in a SOA based systems are widespread, particularly as sensors and smart tags can track users' movements, habits and ongoing preferences.

To the best of our knowledge, The state of the art studies are application specific or domain specific. The scope of existing state of the art models is limited to provide solution for a specific application area. To extend the IoT functionality to an interoperable domain problem, a cross-domain architecture for IoT that will bridges gap between the service oriented paradigm and standard IoT concepts, is presented in this dissertation. The gap has been bridged by dividing the data processing between the low power & low processing capability devices and the cloud based servers.

Performance of proposed architecture is evaluated by integrating applications from three different domains. First application is a sophisticated collaborative context prediction approach based on inductive learning. Experiments are conducted to demonstrate the effectiveness of proposed techniques on real world data sets and virtual data sets. The results show that proposed approach can be applied effectively to diverse application areas.

Second application is forecasting electricity consumption in multi-family residential buildings. Current state of the art approaches for energy management in residential buildings use traditional forecasting techniques based on statistical analysis and machine learning approaches applied on monthly utility meters data, thus restricting hourly forecast. The smart metering infrastructure enables collection of the electricity consumption data at a fine temporal level. Therefore, we applied well recognized machine learning techniques i.e. neural network, regression tree and genetic programming on smart meters data to forecast electricity consumption in residential buildings.

The third application area is multimedia content recommendation in smart home environment. The home network based model is proposed for this application which proactively responds to the user behavior in a smart home. A state of the art learning approach i.e. deep learning is used to enhance the performance of content based collaborative recommendation system. For this purpose, a hierarchical Bayesian model based deep learning is used, which couples a Bayesian formulation of the stacked denoising autoencoder and probabilistic matrix factorization. Experiments on real-world datasets shows that this model can improve the performance of the content recommendation system.

1. Introduction

1.1. Background

The Internet of Things (IoT) has been perceived in multiple ways. A widely accepted definition mentioned in the "Internet of Things, Strategic Research Roadmap" [1] describes IoT as an integrated part of future internet defined as "a dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols". Constituent things have the ability to interact among themselves and the environment, while reacting autonomously to the 'real/physical world' events. These things influence the real world by triggering actions with or without direct human assistance. Interfaces in the form of services facilitate interactions with these 'smart things' over the Internet, taking security and privacy issues into account." In other words, the IoT encompasses all kinds of devices, which use different communication protocols. Its sphere of influence includes a larger and larger number of heterogeneous smart communicating "things". Their management requires highly scalable solutions which are able to overcome their heterogeneity and standards fragmentation.

The major objective of IoT is the development of smart environments/spaces and self-aware entities for climate, food, energy, mobility, digital society and health applications. The developments in smart entities will also encourage advancements in novel technologies needed to address the challenges of public health, aging population, environmental protection, climate change, conservation of energy and scarce materials, enhancements to safety and security and economic prosperity.

The state of the art IoT applications utilize contextual data as an additional resource for performance enhancement[2]–[6]. Recent advancement in the utilization of contextual

information [7]–[10] has given birth to a new dimension of context utilization called context prediction. The context prediction in context aware systems and ubiquitous environment has been one of the important research areas for the last few years. In ubiquitous environment, the context aware systems utilize users' history information to predict their future context which is then used to assist the users in ubiquitous environment. Context prediction focuses on inferring users' context based on analyzing the observed context history. The observed context history is obtained various types of sensors such as GPS, RFID, and a variety of wireless devices. These sensors may provide the context information about users' locations, actions, or the changes in a physical environment of the users. The purpose of context prediction is to predict the subsequent context users will likely enter (if the contexts are locations or situations) or perform (if the contexts are actions). For ubiquitous computing environments, the ability to accurately predict a user's contexts would make it possible to provide context-aware services that are more natural and customized to people's needs.

The recent advances in context prediction open a wide range of possibilities of context-aware computing applications. For instance, a context-prediction application may infer the future location of an office owner and redirect incoming calls to the future location. A context-prediction application may also be useful for enhancing the quality of transportation systems. Based on the information about the current location and location history for a particular user, transportation systems equipped with context prediction technology may be able to assist drivers more effectively by suggesting possible preferred routes. Knowing the current location, time, and calendar of the user, current social situation of the user can also be inferred, such as if the user is in a meeting, in class, waiting in the airport, and the list goes on.

An introduction to the state of the art architectural models and application platforms for IoT offered by standardization organizations, IoT projects, and academia is presented here. Different

architectural models for IoT reflect different perspectives and support different business interests. We believe that the core technological challenges, such as interconnection among heterogeneous devices, very low computational and energy demand need to be overcome for large scale adoption of IoT. Projects in industry and academia around the world strive to solve parts of these challenges. Primary focus is on the development of an open, scalable and trusted architecture. Analyzing the state of the art architectures can help illuminate their strengths and weaknesses. A comparative analysis of these architectures with the proposed architecture is presented here.

1.1.1. IoT architecture standardization

Though many organizations work on the standardization process, we focus here on those that work on IoT. A working group of IEEE deal with the standardization of IoT architecture [11]. The scope of this working group is to define an architectural framework, addressing descriptions of various IoT domains, definitions of IoT domain abstractions, and identification of commonalities between different IoT domains [12]. IEEE is currently considering three tiered architecture of IoT. The prime focus of this architectural model is the connectivity between different IoT devices.

Likewise, ETSI also deals with the architectural view of another perspective of IoT known as M2M. The logical entities in the ETSI proposed architecture are defined as follows:

- M2M Device: A device that runs M2M application(s) using M2M service capabilities.
- M2M Gateway: A gateway that runs M2M application(s) using M2M service capabilities. The gateway acts as a proxy between M2M devices and the network

domain. The M2M gateway may provide services to other connected devices connected which are hidden from the network domain.

- M2M Service Capabilities: Applications that run the service logic and use M2M service capabilities accessible via an open interface.

The oneM2M has a layered model which consists of the application layer, common service layer and network service layer. Application layer is composed of oneM2M applications and related business and operational logic. Common services layer consists of oneM2M service functions that enable oneM2M applications (e.g., management, discovery and policy enforcement). Network services layer provides transport, connectivity and service functions.

1.1.2. State of the art IoT architectures

A scalable and reliable architecture will form the backbone of the future development of IoT. The architecture must cope with the new requirements of IoT and cope with the evolving challenges. We observed that although different architectures for IoT are offered by various stakeholders, yet there is no standardized architecture approved by an authorized body. This lack of standardized architecture contributes to the fuzziness obscuring a clear definition of an IoT system. In this section we will discuss the minimal architectural components of an IoT system.

For this purpose, we have reviewed the suggested IoT architectures offered by various projects, academic and industrial bodies and compared our architectural model with the reference architecture model IoT-A, and two full-fledged architectures developed in the projects BETaaS and OPENIoT.

IoT-A: IoT-A The IoT Architecture Reference Model (ARM) [13] is not an IoT architecture per se, but a set of best practices, guidelines. It can be used as a starting point to

generate specific IoT architectures. It provides an architectural reference model which facilitates the interoperability of IoT systems. It also provides necessary toolsfor the actual integration into the service layer, such as resolution, look up, and discovery of things. The ARM Process defines the steps to generate concrete IoT architectures from business goals. Covered topics include the generation of requirements and their transformation into an architecture using views and perspectives. ARM provides an exhaustive list of so-called Unified Requirements (UNIs) used to generate concrete requirements for a specific architecture. The UNIs are generalized requirements augmented with the views and perspectives of the respective stakeholders. The ARM process makes use of the IoT Reference Model that introduces major IoT concepts such as devices, services, and entities and defines their relations and attributes on an abstract level independent from specific use cases or technologies. Following the established relations, it identifies so-called Functional Groups (FGs) for interacting with instances of the introduced concepts and introduces communication functionalities suitable for heterogeneous IoT settings. Additional features introduced include trust, security, process management, service organization, and more.

BETaaS: Building the Environment for the Things as a Service (BETaaS) [14] defines an actual implementation of the platform besides the overall functionality and architecture. BETaaS consists of a network of gateways ("local cloud of gateways") that seamlessly integrate existing heterogeneous M2M systems. To develop an abstract representation from heterogeneity of the physical layer, BETaaS defines a baseline reference architecture called Things-as-a-Service (TaaS). It provides architectural models for domains, information, communication, security, and functions.

OPENIoT: The OPENIoT research project has defined an architecture utilizing a Sensor Middleware (SM) and a Semantic Directory Service (SDS) [15]. To achieve alignment,

architecture development and specification was based on the Architecture Reference Model (ARM) of IoT-A. The Sensor Middleware (SM) deals with collection, filtering and aggregation of data streams associated with physical and virtual objects. The Cloud Computing Infrastructure (CCI) supports a scalable and elastic storage of data along with their associated metadata. The Semantic Directory Service (SDS) supports registration management and semantic annotation for sensors and services. The Global Scheduler (GS) deals with handling requests for on-demand service deployment and provisioning of access to data sets and services. The Request Definition (RD) element enables the dynamic specification of service requests and the Request Presentation (RP) element takes care of the visualization of the results produced by an executing service. Support for heterogeneous sensor network types and scalability to the proliferation of sensory devices is enabled by supporting a distributed deployment model for SM instances.

A recent academic study focused on data centric IoT application tries to address the cross domain application issue[5]. It introduces a data-centric framework for development of IoT applications executed in clouds. The framework handles connection to data sources, data filtering, and utilization of cloud resources which include provisioning, load balancing, and scheduling, thus enabling developers to focus on the application logic and therefore facilitating the development of IoT applications. Semantic middleware is presented in [16] for managing smart services in IoT environment.

1.1.3. State of the art IoT application platforms

Many IoT platforms have been developed to support IoT application. Main goal of these platforms is the integration of different types of smart objects into the web through RESTful APIs or cloud services.

Table 1.1: Comparison of IoT application platforms.

Platform	Target objects	Service modeling	Service composition	Applications
Axeda [17]	IP networked	Cloud	N/A	Cloud-based platform
BUGswarm [18]	IP networked	RESTful APIs, Cloud	N/A	Cloud-based platform
Carriots [19]	Web-enabled	RESTful APIs	N/A	IoT platform
Etherios [20]	Embedded	Android M2M device	N/A	Platform-as-a-service (PaaS)
EVRYTHNG [21]	Web-enabled	RESTful APIs	Web 2.0 mashup	Personalize/Track/Socialize
GroveStreams [22]	Web-enabled	RESTful APIs	N/A	In-cloud real-time big data analytics for IoT
Nimbits [23]	WSN	RESTful APIs	N/A	In-cloud data processing
Open.Sen.se [24]	(Note specified)	RESTful APIs	Web 2.0 mashup	Data storage Visualization
Paraimpu [25]	Web-enabled	RESTful APIs	Web 2.0 mashup	Social Web of things
NanoService [26]	Mobile phone embedded	Service platform RESTful APIs	N/A	Embedded Web applications
SensorCloud [27]	MicroStrain WSNAndroid, iOS	SensorCloud, OpenData APIs,	N/A	Cloud sensor data storage
ThingSpeak [28]	WSN	RESTful APIs	N/A	Sensor logging, Location tracking, Social network of things
Xively (Pachube) [29]	(Not specified)	RESTful APIs Sockets, MQTT	N/A	IoT public cloud Platform as a service (PaaS)
Yaler [30]	Embedded, (Arduino, BeagleBoneNe tduino, Raspberry Pi)	RESTful APIs SSH Service	N/A	Relay infrastructure for Web access of devices
Proposed CIoT	Web enabled, Mobile phone, embedded	Cloud, Restful APIs	Integrated	IoT platform, cloud, personalized services, collaborative environments

These platforms provide mid-point services to encompass underlying heterogeneous smart objects into Web interfaces that can further integrate into modern Web infrastructures such as cloud and platform-as-a-service (PaaS). These approaches expose some scalability issues of IoT systems since each platform faces its own routing discrepancy and protocol translation. RESTful APIs make service composition possible in the form of mashups; even though, there are only few of them [21][31]. Besides, these platforms still fail to answer how multiple smart objects can interact. In other words, Service composition models, context prediction models and application collaboration models are neither reported in literature nor developed in practical.

1.2. Problem statement

IoT requires devices and applications that can easily connect and exchange information in an ad-hoc fashion with other systems. This will require devices and services to express needs and capabilities in formalized ways. To facilitate the interoperability in the IoT, further research into data-centric technologies is needed. Examples of challenges are large-scale distributed data processing, new approaches to enhance the efficiency of IoT through services collaboration, rule engines and approaches for hybrid reasoning overlarge heterogeneous data, and semantic-based discovery of devices. The effectiveness of the IoT services largely depends on the performance, trustworthiness and reliability of the services. Current state of the art solutions/approaches of the IoT applications exploring data-centric technologies for the processing of contextual data gathered to ubiquitous sensors in a smart environment. However, the performance remains a challenge with the existing approaches.

To enhance the performance of the IoT services, this work proposes a solution to address the issue of context prediction in a multiple users smart environment through inductive learning and predictive rule mining approach. The contribution of this work is twofold i.e. a collaborative

context prediction in smart IoT environment and the integration of IoT architecture with service oriented architecture. Besides, this work investigated the applicability of the machine learning approaches to another challenging problem i.e. electricity consumption forecasting in a multi-family residential buildings.

1.3. Contribution

The Contribution of this work is many-fold i.e. enhance the performance of the state of the art data processing techniques, investigate the applicability of the state of the art machine learning techniques to a new challenging tasks, and propose a unified architecture for the heterogeneous IoT applications. This unified architecture is designed to integrate application from different domains i.e. Human activity prediction, energy consumption forecasting, smart home and smart service composition. Besides, the integrated smart architecture, this work also proposes a novel context prediction approach which covers a) the collaboration among the user for context sharing, and (b) the selective collaboration among the similar kinds of users. To the best of our knowledge, this approach has not been reported in the literature. The advantage of the profile based collaboration is that it is simple to implement, secure and more efficient due to additional information. Also we proposes a novel rule base service composition mechanism for the IoT.

Besides, this work investigate and analyses the electricity consumption forecasting in multi-family residential buildings at various spatial and temporal granularity level using machine learning approaches i.e. neural network, classification & regression tree and genetic programming and the impact of different preprocessing techniques on the performance of these approaches. To the best of our knowledge, these machine learning approaches has never been utilized for the electricity consumption forecasting in multi-family residential buildings.

Furthermore, due to widespread applications of deep learning approaches, this work proposes a deep learning algorithm for the collaborative filtering based content recommendation in the integrated IoT system.

2. Related work

The Internet of Things (IoT) has attracted a lot of attention from the academia and industry. It is believed to have enabled the Internet to reach out into the physical world of Internet-connected devices [32][33]. The IoT, as an emerging concept alongside the recent wave of technological advancements, refers to the connection of various physical objects in real life through wireless tags and sensors over network protocols similar to those used in the Internet [34],thus giving birth to a new breed of devices commonly referred to as smart objects which can become part of the existing Internet. Built on the IoT, the physical world will become an intelligent hyper-connected world with smart physical objects tagged wirelessly, bringing many science fiction stories and scenarios to life[35][36]. The recent development of Google Glass and Apple's iWatch are brilliant demonstrations of this new technological trend. In the future, the existing Internet will become the backbone network for major data and information transfer where most objects in real life will be linked together pervasively [37].

Extended from the IoT, the concepts of smart home, smart community, smart city [36], and even the smart planet promoted by IBM suddenly become foreseeable in near future [37]. Advances in wireless networks and data processing have facilitated transformation of traditional Internet into an intelligent IoT which is capable of interconnecting diverse "things" into the physical world [38], [39]. In practice, the inexpensive intelligent sensor networks, radio-frequency identification (RFID) tags, and wireless devices are widely used to collect data, making information exchange and processing feasible[40]–[42]. This further leads to changes in operations of many existing enterprise systems and decision support systems [37]. In the foreseeable future, business processes and models will also be changed and adapted to the IoT paradigm accordingly [1], [43], [44].

In the past few years, the IoT has attracted a lot of attention from researchers and achieved significant growth [35], [35], [45], [46]. One concern lies in the communication and interaction process among different devices. The architecture of the IoT conceptually consists of three layers: sensing layer where many wireless sensors are located, network layer where data collected from sensors is transmitted and processed, and application layer where various applications access the functions and information provided by sensing and network layers[40]. Different wireless devices might use different protocols with different object identification, data representation, and data transmission formats, raising the issue of coherent processing information from multiple heterogeneous resources.

To tackle this issue, researchers have proposed service oriented architectures (SOAs), built on top of the network layer so that data processing can be easily handled through different service components [38]–[40], [47], [48]. In the SOA of IoT, interaction with and operations of different wireless devices are treated as different service components, making it possible for application layer software to access resources exposed by devices as services. These services are defined and classified based on real-world services, directly derived by physical resources. Services are capable of sensing, processing data and operating device entities either by providing interaction interfaces or by generating events. Therefore the service oriented IoT can control, manage, and interact with the real world by means of "services," which enable bi-directional user-to-object information exchange and interaction [3], [16].

Existing research on service-oriented IoT have strived to contribute either from the architecture deployment perspective or the service classification, interaction, and discovery perspective. For example, Organero et al. [48] proposed a service-oriented platform for a personalized e-learning environment involving web 2.0-related service at an open and personal framework, while Vinoski[40] introduced several mid-level architectures in an IoT context,

including a data collection, data mapping and service encapsulation model. Liu et al. [38] investigated IoT-based mobile service deployments in support of pervasive computing paradigm. Guinard et al. [47] proposed a service-oriented platform for IoT, involving a large number of service operations, such as service discovery, query, classification, provision, and so on. The related services in the IoT can constitute a complicated service, where a service is handled as a modular and adaptive middleware component. Butt et al. [45] proposed service discovery architecture for the IoT and its accompanying RESTful protocol. Their work addresses IoT deployments with severe constraints in device processing power and network bandwidth [45], [49]. Furthermore, architecture model for a service-oriented IoT is investigated in [47], where the author proposed a method to model collaborative virtual objects architecture via a generic interaction between services.

To the best of our knowledge, the aforementioned approaches lack two very important factors in the IoT domain i.e. context prediction and service level agreement for collaboration. Current state of the art architecture inherits the very same limitations. Although privacy concerns in collaborative data sharing is a hurdle, but due to its importance to the components, its applicability needs to be investigated in the new paradigm. Application level collaboration is considered as distributed consensus decision making for services over IoT, which is of high importance in a context of resource-constrained wireless devices. Because the processing power and storage capacity of wireless devices in IoT is rather limited, there is a high demand for the discovery and coordination of services to efficiently process data over the IoT. Therefore, following challenges needs to be addressed in current service oriented IoT [3], [16], [17].

- The IoT should be able to provide users with services for sensing information of interest, which might involve multiple interconnected IoT edge devices. This imposes a challenge on efficient data propagation.

- The IoT should be able to run distributed decision making process for service detection, classification, composition, and data processing, on demand.
- Services should be able to cooperate for tasks completion. Information consensus between services should guarantee that each service share task-critical information over the IoT.

This work aims to solve mentioned challenges by proposing a collaborative architecture based on distributed algorithm for decision making at edge services in the service-oriented IoT. Specifically, the main contributions are summarized as follows.

1) A service provision framework is proposed, which deals with the representation, discovery, detection, and composition of services.

2) A distributed decision making method for service composition is proposed which effectively select suitable services according to application layer requirements.

3) A distributed algorithm is proposed for robust decision when multiple services are required to interact with each other.

3. Collaborative IoT architecture (Service-oriented IoT architecture)

In this chapter, a context prediction based collaborative architecture is presented for service oriented IoT; called as Collaborative IoT (CIoT) architecture . The proposed CIoT architecture is the extension and amalgamation of the two IoT architectural structures i.e. service oriented IoT [50] and IoT reference model [51], however, the capabilities of both the architectural concepts are enhanced with the additional components i.e. context prediction [8] and service collaboration through service level agreement. The context prediction component's prime concentration is on the prediction of the situation for certain services. The collaboration component deals with the collaboration of various applications and services among each other. CIoT consist of four layers i.e. acquisition & control layer, processing layer, service layer, and application layer as illustrated in Figure 3.1. Layered wise component description of the architecture is given below.

3.1. Application layer

Application layer is the top layer of the architecture which deals with users interaction, handle the environmental level details and proactively adapt the environment according to the user's context and the system's recommendation. This layer define the application scenario, resources, data sharing strategies, collaboration mechanism, authorization, user interface and medium of communication. When a new application needs to be integrated into the CIoT, all these information in the form Resources Description (RD) needs to be explicitly defined. The inputs and outputs of the application also needs to be explicitly described. Also, the output format of the application/service i.e. notifications (push/pull), reports, type of visualization is defined. The proposed CIoT is evaluated with an integrated environment composed of three

applications as illustrated in Figure 3.1. These applications are activity prediction in smart office through user profile based collaboration, entertainment enrichment in smart homes through context based multimedia content recommendation, and energy consumption forecasting in smart buildings. The idea behind the selection of these applications is their immense importance and research potentials. Also, these applications belongs to diverse real life domains which is an appropriate case to evaluate the applicability of the proposed architecture. The application selection procedure does not carry any parametric evaluation of the selected applications, but are randomly selected from the pool of available application scenarios. Therefore, this thesis does not consider application level clustering and strictly focuses on the services level clustering and collaboration. The detail discussion on the selected applications is carried out in their respective chapters.

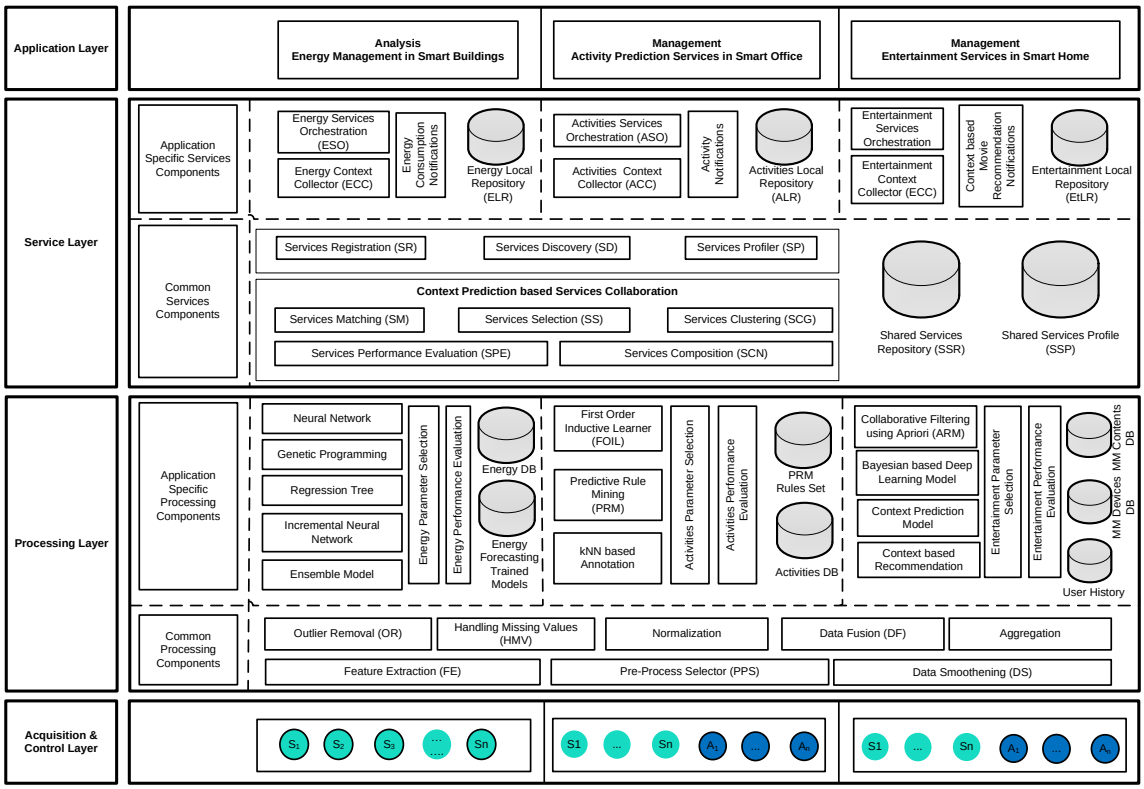


Figure 3.1: Layered component architecture of the CIoT

3.2. Service layer

The service layer is the core layer of the proposed CIoT architecture which handles all the services related functionalities. A service is considered as a collection of data and associated behaviors to accomplish a particular function or feature of a device or portions of a device. Keeping in view a service as the basic building block of the CIoT and considering the Thing as a Service (TaaS) and Process as a Service (PaaS), this layer describes the service registration, discovery, orchestration, composition and collaboration aspects in the service-oriented fashion. Services in CIoT are created and deployed according to the following steps [18]: 1) development of service composition & collaboration platform; 2) abstraction of communication capabilities & devices functionalities; and 3) provision of a common set of services. In these phases, a services identification process is involved for context based services collaboration and services clustering, with which a trust mechanism is devised in the CIoT among services.

As mentioned in [13], "*a service may refer other primary or secondary services and/or a set of characteristics that make up the service*". Based on this definition, services in CIoT are categorized into two types: primary and secondary. The former denotes services that expose the primary functionalities at an IoT node, which can be seen as the basic component of a service and can be invoked by another service. A secondary service can enhance a primary or other secondary services by providing auxiliary functionality. A service may consist of one or more characteristics, such as service data structure, permissions, descriptors, and other attributes. When service interact with each other and share information, the inherent problem of privacy in sharing of data & resources is handled using the following procedure: The services are divided into two categories i.e. local & shared. Local services are internal to the application and can only be used by the base application for orchestration and sharing data with other applications, while the shared services are used for composition and collaboration among other applications. The

private information are utilized only by the local services and are not shared among the other applications unless explicity authorized by the service owners. The service definition includes the access rights to the assoicated resources (sensors, actuators), which can be shared by the service provider. The public services shares only abstract level of information and do use anonymize data.

Figure 3.2 shows the flow model of the processing of the service layer. The component's interaction is illustrated with their dependencies on each other. when a task request is initiated to the CIoT, the service handling mechanism evaluate the request and execute the concerned service. When a service orchestration is required, the orchestration unit is invoked and the aggregated services are formed for the requested task as: if $S_1, \dots S_m$ are disjoint services, then

$$|S_1 \cup S_2 \cup \dots \cup S_m| = |S_1| + |S_2| + \cdots + |S_m|$$

To write this without the “dots” that indicate left-out material, we write

$$\left|\bigcup_{i=1}^{m} S_i\right| = \sum_{i=1}^{m} |S_i|$$

The service selection mechanism, when select the service (atomic/composite) collaborate with the other unit of the system for complex processing. When a predictive reasoning is required for a task, the processing component collaborate with the processing layer and execute the task upon the implicit or explicit requests of the users.

In this work, orchestration and composition are considered as two separate entities i.e. Orchestration is the integration of unit services of the same application domain while compsition is the integration of cross domain services. One of the core concept of the service layers is context based collaboration which handle the connection between objects and sharing of resources & data. For example, the prediction object is connected with the recommendation

object which enhances the performance of the notification recommendation. The notification recommendation of the recommendation component at the service layer is based on the context prediction module at the processing layer.

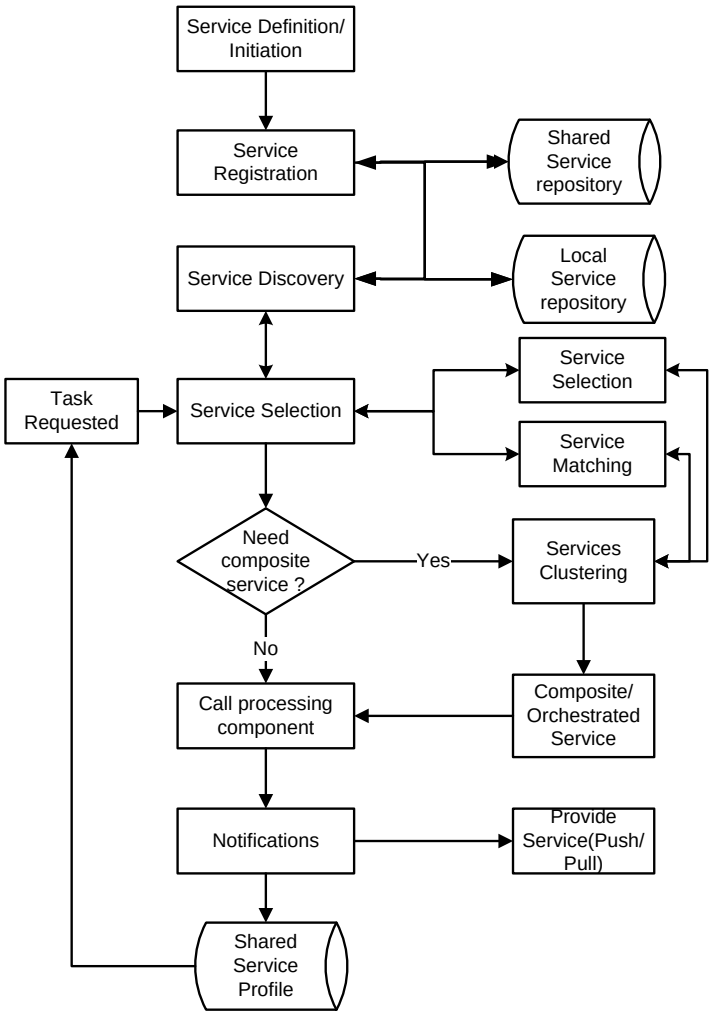


Figure 3.2: Service layer flow model

The components of the service layer is divided into two groups i.e. application specific services components and common services components. Detail description of each component is presented as below:

3.2.1. Common Services Components

The components which are common among all the applications at the service layer are called as common services components. These components are used by all the applications and services registerd in the proposed CIoT. These components are elaborated as follows:

3.2.1.1. Service Registration (SR)

Services registration is the process of registering services in the system. The application specific services are stored in the application specific local repositories while the shared services are registered in the Shared Services Repository (SSR). All the services are registered with the following properties:

- ID
- Name
- Keywords
- Resources
- Accessing rights
- Visibility
- Location

While registering a new service into the CIoT, the service visibility needs to be explicitly mentioned. When a service visibility is set to "shared" then that service can be used for service composition by other application domains. When the service visibility is set to "local" then that service can only be used by that application and also can be used for service orchestration at the application level. When a service joins the CIoT, it advertises its capabilities by broadcasting the service functionalities it can provide. Upon receiving a message, the existing services can get these new services as IoT services by exchanging service information [14]. In doing so, a pool of

registered services are visible to each other and can benefits from the potentials of each other. The dynamic nature of the IoT changes the structure of the services hierarchy change frequently.

3.2.1.2. Service Discovery (SD)

Service discovery is the process of finding the registered services. This work devise a smart quering mechanism to discover the registered services. Shared services are located by the smart integrated querying mechanism in the SSR while the private devices are only visible to the respective application's querying mechanism in their respective local repositories i.e. ELR, HLR, EtLR. This will restrict the visibility of local services from the other applications. The registered services $S = \{s_1, s_1, \dots, s_n, \}$ are associated with each application mentioned in the CIoT. When the users wants to find a specific service, they sends their query for a service through registered keywords. The user query is entertained by the SD module and locate the service in SSR and Local Repositories(LRs) using the assumption of equation 3.1. Once the registered services are located, the detail is provided to the clients.

$$find(s) = (\forall\, s \in S\, \exists\, SD\ in\ repository) \tag{3.1}$$

3.2.1.3. Service Profiler (SP)

Service profiler is the process of collecting the information about the services activities. When the services are local services then the profiler unit store the collected information in the respective local repository while for the shared services, the information is stored in the shard services profiles (SSP) repository. We collected the following information about the services.

- Accuracy: The number of successful tasks completed by a service.

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N} C_T/N \tag{3.2}$$

where N is the total number of tasks requested to the service, C_T is the number of successfully completed tasks.

- Response: The response of the service when consulted for a specific task.

$$weight_{response} = \sum_{i=0}^{N} {(PR - NR)}/{N} \quad (3.3)$$

where N is the total number of responses, PR is positive responses and NR is negative responses.

- Capacity: No of request can handle at time t

$$Max_t(request) \quad (3.4)$$

- Processing time: Cost based on size of data

$$s = size(i)$$
$$Z_s = \sum(r_s, p_s, d_s) \quad (3.5)$$

where r_s is the data receiving time, p_s is the data processing time, and d_s is the response delivery time.

- Location: The location of service in which a specific task is accomplished.
- Resources: The resources utilized for accomplishing a specific task.

3.2.1.4. Service Matching (SM)

Service matching is the process of evaluating the services to incorporate into the system. When a service is the discovered by the SD module and is available for multiple applications, then a distributed process is needed to get it efficiently combined into the proposed CIoT. In this work, we define a normalized matching value x_{match} to evaluate the matching of the new coming s_1 with the existing services $K(K \geq 1)$ based on multi-attributes as defined in the SP section above.

$$X_{matc\,h}(s_i, s_j) = match(s_i, s_j), j = 1, \dots, K \tag{3.6}$$

in which the $match(s_i, s_j)$ function is used to evaluate the similarity of services.

A new service can become part of a service pool when an agreement is achieved for all involved IoT services. The proposed matching value-based method allows one to find the possibility that the services can be composed by existing IoT services [52], [53]. Therefore, matching values at nodes are gathered and synthesized to reach a final decision acceptable by all. Through agreement-based decision making, services in IoT are not only working to achieve better solutions, but also to promote trust [54], [55].

The Services IoT system is modeled by a graph of services and connections i.e. G=(S, C). Let $x_s(t)$ represent the state value (used to evaluate the matching value for new incoming services) at service s at time t , which can be intuitively understood as the estimate of the consensus value of s. At each node s ($s \in$ S), let $x_s(0)$ denote the initial measured value at s, which can be further updated through iterative exchanges between neighbor services and the agreement or averaging can be achieved at all the nodes:

$$x_s(t+1) = x_s(t) - \mu \sum_{u \in \mathbb{N}_s} (x_s(t) - x_u(t)) \tag{3.7}$$

where $\mathbb{N}_s$ denotes the neighbor list of s and μ denotes the step size in each iteration. The convergence properties of this equation is largely determined by the Laplacian matrix $\boldsymbol{L}$, thus

$$L(t+1) = f(x) = \begin{cases} d_s, & u = s \\ -1 & (u, v) \in C \\ 0 & else \end{cases} \tag{3.8}$$

in which d_s is the degree at service s. Let an $n \times 1$ vector $x(t)$ denote the states vector of all nodes at time t, then equation 3.7 can be formatted as

$$x(t+1) = x(t) - \mu L x(t) = (I - \mu L) x(t) \tag{3.9}$$

It can be rewritten as

$$x(t+1) = Wx(t) \tag{3.10}$$

in which $W = I - \mu L$.

3.2.1.5. Service Selection (SS)

Service selection is the process of selecting an atomic service or make a composite/orchestrated service for a certain task. When a task is received from the CIoT client, the service selection module is invoked to select a specific service to perform that task. The service selection calls upon the service matching module and finds which services can perform the task at hand. When the service selection module finds that the service composition or orchestration is required to perform the task then the task and selected services are forwarded to clustering module which cluster the services bases on the similarity index of the services and task.

3.2.1.6. Services Clustering (SCG)

IoT applications can be easily grouped into multiple clusters by well-known clustering algorithms, such as FCM (fuzzy C-Means), kNN(k nearest neighbor) location-basedclustering, [22]–[25], etc. Since this work has considered very few application domain for evaluation, therefore, an application level clustering is not carried out. However, services are clustered using FCM and the deployment has be broken down into clusters according to application requirements as mentioned in SM above. In each cluster, a service is selected as cluster head (CH) and is able to exchange information with other CHs. In a deployment of K clusters $\{C_1, C_2, \dots, C_k\}$, each cluster C_i consists $|C_i|$ nodes. For a cluster C_k, the agreements are formulated as:

$$x_k(t) = (I - \mu_k L)x_k \qquad k = 1, \dots, K.$$

At each CH, the iterative averaging problem is concurrently solved and for each cluster a local agreement is achieved as

$$x_k(t) = (W)_k^t x_k(0), \qquad k = 1, \dots, K \tag{3.11}$$

here we have

$$\lim_{t\to\infty} x_k(t) = \lim_{t\to\infty} \qquad W_k(t) = x_k(0)1, \quad k = 1, \dots, K \tag{3.12}$$

in which **1** denotes the vector with all coefficients one. According to [56], the convergence rate can be measured by

$$\rho\left(W_k - \frac{11^T}{|C_k|}\right) \tag{3.13}$$

and the associated convergence time is

$$\tau = \frac{1}{\log(\frac{1}{\rho})} \tag{3.14}$$

The *K* concurrent subproblems could be solved by

$$\min\left(\rho\left(W_k - \frac{11^T}{|C_k|}\right)\right) \tag{3.15a}$$

$$s.t. \lim_{t\to\infty} W_k = \frac{11^T}{n} \tag{3.15b}$$

$$W_k 1 = 1. \tag{3.15c}$$

Similarly, an agreement for all the services involved is obtained as $x_g = [x_1, x_2, \dots, x_K]^T$

$$X_g(\tau) = W_g^\tau X_g(0) \tag{3.16}$$

in which $x_g(0) = [x_1(t), x_2(t), \dots, x_K(t)]^T$. The equation 3.15 can be solved as

$$\min\left(\rho\left(W_g - \frac{11^T}{|C_g|}\right)\right) \tag{3.17a}$$

$$s.t. \lim_{t\to\infty} W_g = \frac{11^T}{n} \tag{3.17b}$$

$$W_g 1 = 1. \tag{3.17c}$$

The convergence time of the whole collaboration is calculated as

$$\tau = \max_{k=1,\dots,K}(\tau_k) + \tau_g \tag{3.18}$$

Service level collaboration is carried out among the unit services and components of the architecture while the application level collaboration is carried out at the cross domain application level.

3.2.1.7. Services Performance Evaluation (SPE)

Services performance evaluation means the perfomance of a service over a certain period of time as an atomic service or as a part of a composite service. The follwong parameters (mentioned in SM section) are used to evaluate the performance of the service:

- Processing time: Cost based on size of data

$$s = size(i)$$

$$Z_s = \sum(r_s, p_s, d_s) \tag{3.19}$$

- Accuracy: The number of successful tasks completed by a service.

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N} C_T/N \tag{3.20}$$

where N is the total number of tasks requested to the service, C_T is the number of successfully completed tasks.

- Response: The response of the service when consulted for a specific task.

$$weight_{response} = \sum_{i=0}^{N} {(PR - NR)}/{N} \quad (3.21)$$

where N is the total number of responses, PR is positive responses and NR is negative responses.

3.2.1.8. Shared Services Repository (SSR)

This repository contains the information about the shared services and their associated resources. When a service or a resource is declared as a shared service/resource then it is stored in this repository. Besides, when a service discovery module searches for a service, it not only search the local repositories but also this repository. This repository does not impose any constraints on the services utilization or resources access from within the registered applications. The data stored in this repository is visible to all the registered applications and their respective service. The utilization of the services is based on the matching index as described in SM module and SCG module. The following table shows the attributes of the SSR.

ServiceID	Name	Accessing rights	Visibility	Location	Associated applications	Associated resources

3.2.1.9. Shared Services Profile (SSP)

This repository contains the profile information of the services registered in the SSR. The services profile consists of services utilization by a certain application for a certain tasks, service utilization time, service utilization with other services and the trust worthiness of the service with other services.

ServiceID	Name	Utilized by	Utilized at	Task accomplished	Used in consultation	Trust index with other services

3.2.1.10. Service Composition (SCN)

The process of integrating multiple atomic service to perform a complex task jointly is considered as service composition. Service composition is carried out in two ways i.e. automatic and static. In automatic services composition, the system selects multiple services to make a composite service for a certain task while in static service composition the system designer knows the dependencies of the service on each other and based on his experience, he makes the composite service for certain tasks. For example, weather service is composed of multiple service i.e. temperature service, humidity service, air quality service and rain prediction service.

For this work, a hybrid approach for service composition is devised using both automatic and static approaches. The automatic composition approach is derived using the SCG, SM, SS modules and is discussed in the respective sections, however, a static approach which is a rules based composition in which the user selects the services to compose on run time and the system make a composite service of the selected services.

The automatic composition is carried out as follows: when a service is well classified, it can be properly integrated into the IoT to make composite services. Since, this work considers service composition across the applications therefore, the composition process needs an agreement among involved applications. The agreement value is calculated using SM module and SCG module. For one application, the discovered service s_1 can be combined with existing service(s) to form a complex service s_2 (composite service) according to the following conditions:

$$s_1.output \cong s_2.input \tag{3.22}$$

$$\xi_s(s_2.output) \succ \xi_s(s_1.input) \tag{3.23}$$

The first condition ensures that the output of service s_1 can be consumed by s_2 . The second condition states that the output of s_1 can be completely accepted by s_2.

When a decision is made by multiple services, it is important for the services to reach an agreement which is known as Consenses Decision Making (CDM) [48]. In CDM, the input and ideas of all IoT end services are gathered and synthesized to arrive at a final decision acceptable by all. The following assumptions are made regarding the consenses of collaboration among the involoved services and is mathematically presented in SM and SCG modules.

- Through consensus, a better solution can be achieved and the trustiness between applications can be promoted.
- The services being merged are in similar domains where a quantitative criterion can be used to evaluate the composition's feasibility.

Normally, two kinds of agreements situations are involved in the service-oriented IoT [57].

- Data agreement, whereby multiple applications must reach to an agreement when referring to the same piece of data. For example, when temperature at a specific location

can be provided by multiple IoT services/nodes, then a data consensus will increase confidence on the measurement result.

- Service agreement, which helps the IoT to build composition services with multiple services provided by different IoT applications.

Rule based service composition is carried out explicitly by users utilizing his domain knowledge. The detail mechanism of rules based service composition is given below:

3.2.1.11. Rules based service composition

Figure 3.3 shows the architecture of our proposed rules based service composition system. The system composed of service description, service profile, and rules set for each service. The two services are added together to form a composite service. The composite service is made based on the composition rules. These are user defined rules and hence considered as static composition.

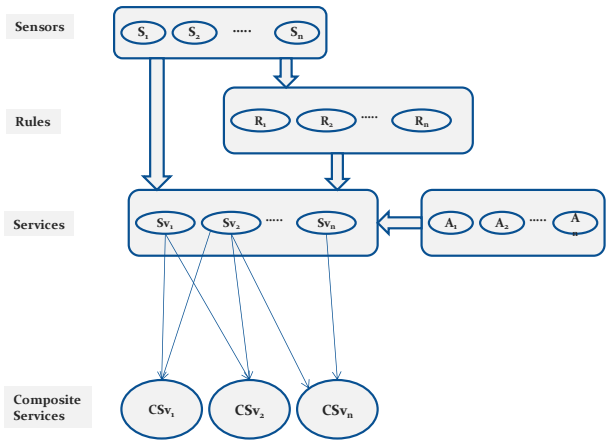


Figure 3.3: Static service composition architecture

An XML based service description mechanism is devised for rule based service composition. The equation shows the structure of a service which is composed of four objects

i.e. associated sensors, service definition, rules definition, and associated actuators and is defined as:

$$SD_x = \begin{cases} ASen_x & x = 1, \dots, k \\ SD_x & x = 1, \dots, k \\ AAct_x & x = 1, \dots, k \end{cases} \quad (3.24)$$

- **Sensors profile:** The sensors profile means the sensors which are registered in the system and are providing the data to the defined service. The sensor profile keeps the basic information about the registered sensors in the system such as Id, name, type, latitude, longitude, absolute (abs) address, and relative (rel) address. The sensors are binded with service along with actuators. Every service is binded to different number of sensors. For example, service A is controlling the temperature of the room A, which will require the temperature sensors in the room to send data to the service. Besides the sensor's configuration file will be saved at the application server so that all the registered sensors are visible to the application server and can be used during the service configuration and service composition.
- **Service profile:** The service profile refers to the basic information of the service. The service definition part consists of basic information and addresses of the service host. The ID, Name, and Keywords attributes are useful for discovery/findability the service and the IP, Port, and URL can be used for subscription of the service. However the Absolute (Abs) address and Relative (Rel) addresses are the physical or area of the service. These addresses can be used for location/area specific services, which can restrict subscription to the service in a specific geographical area.
- **Rules definition:** The important part of the service description is the service behavior, which is defined through the rules set associated with the service. The rule set defined

by the user or some artificial intelligence mechanism is stored in the service profile. The service uses this rule set for decision in the service execution.

- **Actuators profile:** Associated actuators refer to the actuators which are registered in the system and are associated with the services. When rules are triggers for an event, the system take some action either through the associated actuators or recommend other conclusions as defined in the rules set.
- **Service generation mechanism:** The service generation mechanism is shown as a flow diagram in Figure 3.5. The figure shows the dependencies of the proposed system's units on each other. The sensors profile along with service definition is used to make a rules set for pattern identification and activity recognition. Based on the recognized activity and service rules the actuator control rule set is defined.

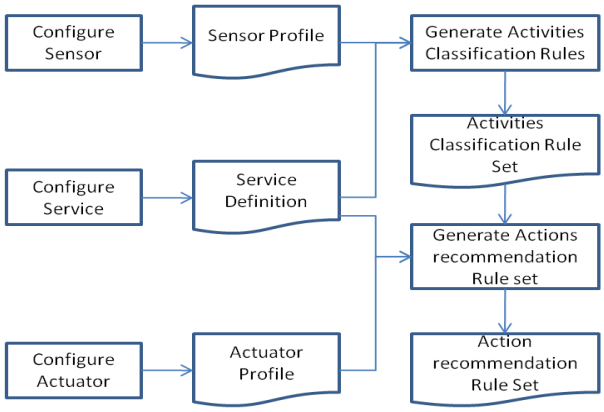


Figure 3.4: Atomic service generation mechanism

- **Activity classification:** The important part of the service generation mechanism is activity classification. The activity classification is based on the association rules. The rules comprised of pre-condition and post-condition as follow:

$$\text{If } \{s_1=x_1, s_2=x_2 \ldots\ldots\ldots\ldots s_3=x\} \text{ Then } \{a=y\} \tag{3.25}$$

The rule set is created manually by the administrator of the system using activities rules section of the prototype. The rules are generated based on threshold values of the sensors and class label for activities.

The Figure 3.5 shows the rules generation mechanism, which is based on sensors, and services. When the user initializes the classification module, the system looks for the registered sensors, and services. If any of dependent module is missing, then the system give message and exit the module. And when all the modules are configured then the system allows the user to configure a new rule. When the user configure a new rule, the system checks the rules set file, if it exists then the new rule is appended otherwise it create a new rule set and save the rule in this rule set.

Figure 3.6 shows the utilization of the activities rule set. When the system is initialized for activity classification, it is presented with a pattern of sensors values. The system checks the pattern in the classifier's rule set which is stored as an xml file. If the system finds the pattern then it checks the values to match the condition. When the pattern values matches the rule, the system triggers that rule, and assign the class label otherwise a pattern miss message is issued and the pattern is stored in a temporary file. The system keeps track of the miss patterns, if the number of miss patterns increases a certain threshold values, then the system initiates a message to the administrator to assign a class label or discard the patterns. The system administrator then decides whether to discard the patterns or assign a new class label to that pattern.

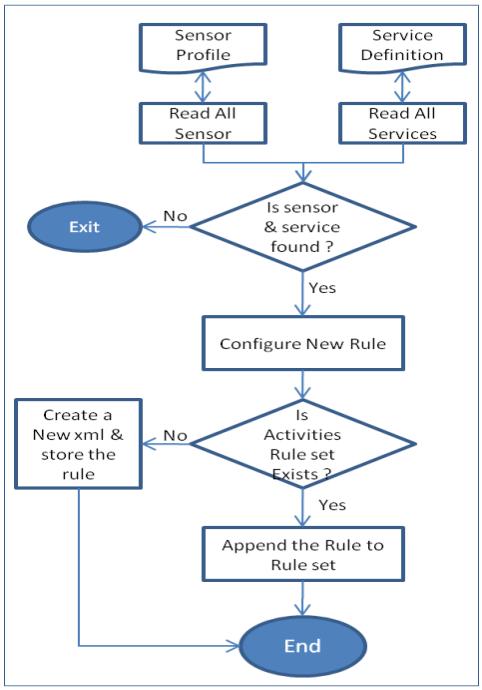


Figure 3.5: Activity rules generation flow diagram

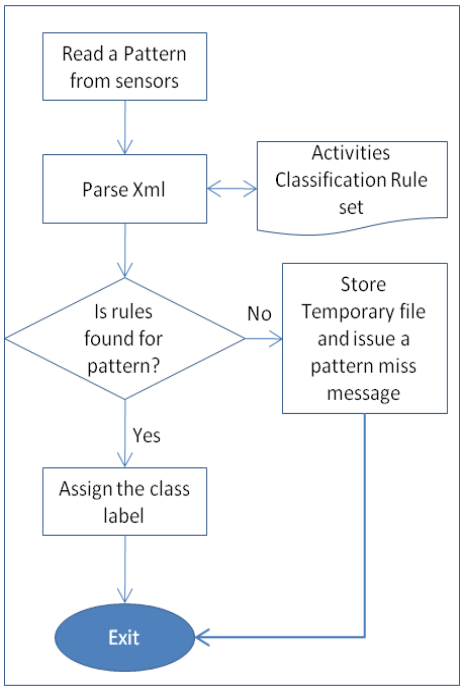


Figure 3.6: Rules utilization flow diagram

3.2.2. Application Specific Services Components

The application specific components as utilized in the experiments are described as below:

3.2.2.1. Activities Context Collector (ACC)

This component is collecting the contextual information of the activities. activities are usually overlapped in a multi-user environment, therefore, this component play a vital role in understanding the activity and enhancing the prediction performance. The contextual information collected by this component is stored in the ALR & SSP. This component ensure protecting to expose the services information or the personal information of the users to the SSP. The personal information is collected by this component which is only stored in the ALR and not

in SSP which ensure to preserve the privacy of the users. The following data is collected and stored by this component.

ServiceID	Name	Utilized by	Utilized at	Task accomplished	Used in consultation	Trust index with other services	Service consumed by Person ID	Service consumed at

3.2.2.2. Activity Notifications

This component generate the notifications based on the predicted activity in a specific context. It also considers the user preferences in order to provide notifications. Both the push and pull notification mechanism is adopted for the ease of the users. The pull notification requires an explicit request from the user while the push notifications need the user's frequency preferences. Based on the user's set frequency, this module sends the notifications to the preferred device.

3.2.2.3. Activities Services Orchestration (ASO)

Service orchestration is the integration/communication of different unit services of the application to perform a complex task. A unit service is considered as a small modules which perform a specific task. For example, temperature monitoring, smoke detector, air quality monitoring. Normally fewer resources are associated with a unit service. An application consist of multiple dependent services which are further clustered based on their natural dependencies, behavior or usability explicity by the application developers or implicity by the smart application. For example, weather forecasting service is naturally dependent on temperature monitoring service and rain monitoring service.

The services defined for this experiment are: Accelerometer service, Gyroscope service, Motion service, RFID service, Temperature service, Wind Speed service, Humidity service,

Location service, Actuator control service, kNN service, FOIL service, PRM service, Context prediction service, and Performance evaluation service. The orchestration is carried out from these services which enables to perform a complex tasks such as annotation of the user activities and notification of the activities in smart office. Annotation service is an orchestrated service based on Accelerometer service and Gyroscope service.

$$Annotation(s) = integration\begin{cases} Accelerometer\ service \\ Gyroscope\ service \\ kNN\ service \end{cases} \tag{3.26}$$

Similarly environment control service is orchestrated from Temperature service, Wind Speed service, Humidity service, and Actuator control service.

3.2.2.4. Activities Local Repository (ALP)

This repository keeps the local services information about the activities in smart office. The following table is maintained in this repository:

ServiceID	Name	Utilized by	Utilized at	Task accomplished	Used in consultation	Trust index with other services	Service consumed by Person ID	Service consumed at

The ServiceID is the id of the registered service and Name is the name of the registered service. A unique id is assigned to each service which enables the other components like orchestration and collaboration to interact with the service using the unique id. The "utilized by" attribute keep track of the applications which invoke the service. This helps in keeping the user's profile and maintaining the history of the users utilizing the service. "Utilized at" property means that the service is invoked at time t. This property helps in maintaining the history of the services utilization with respect to time. "Task accomplished" property maintains the history of the

service by the tasks it successfully completed; By task completion we means the queries successfully entertained. The "user in consultation" property keeps the record of the service orchestration which enables the orchestration component to build a trust mechanism for the service when an orchestration is need in future. The trust index increases as the number of times successful orchestration is carried out among the services. The "trust index with other services" shows the trust index of the service with other services. " Service consumed by Person ID " property keeps track of the persons who utilizes the service. The property " Service consumed at " keeps track of the location from which the service is invoked.

3.2.2.5. Energy Context Collector

This component is collecting the contextual information of the energy consumption. energy consumption is heavily dependent on the user behavior and environmental factors, therefore, this component play a vital role in the efficient forecasting of electricity consumption and enhancing the forecasting model's performance which further leads to a small MAPE and high reliability index. The contextual information collected by this component is stored in the ELR & SSP. This component ensure protecting to expose the services information or the personal information of the users to the SSP. The personal information is collected by this component which is only stored in the ELR and not in SSP which ensure to preserve the privacy of the users in a multi-family residential building. The following data is collected and stored by this component.

ServiceID	Name	Utilized by	Utilized at	Task accomplished	Used in consultation	Trust index with other services	Service consumed by Person ID	Service consumed at

3.2.2.6. Energy Consumption Notifications

This component generate the notifications based on the forecasted results of the models implemented at the processing layer. It also considers the user preferences in order to provide notifications. Both the push and pull notification mechanism is adopted for the ease of the users. The pull notification requires an explicit request from the user while the push notifications need the user's frequency preferences. Based on the user's set frequency, this module sends the notifications to the preferred device.

3.2.2.7. Energy Services Orchestration (ESO)

Service orchestration is the process of combining multiple service to perform a complex task. The services defined for this experiment are: Smart meter service, Temporal service, Temperature service, and Humidity service, Actuator control service, NN service, RTREE service, GP service, IL service, EM service, and Performance evaluation service. The orchestration is carried out from these services which enables to perform a complex tasks such as weather analyzer and notification of the energy consumption in smart building.

The notification service is a customized service work both in push and pull notification style. Based on the user preferences and alert level, the notification is decided. Besides, these notification are coordinated with the context prediction unit which avoids any notification of disturbance. A blind notification system leads to annoy the users and some time leads to stop the notification services. When the users stop the notification services, sometime an important notification is missed. Therefore, the smart context based notification is devised based on context of the user, which avoid annoying situations.

3.2.2.8. Energy Local Repository

This repository keeps the local services information about the activities in smart office. The following table is maintained in this repository: The ServiceID is the id of the registered service and Name is the name of the registered service. A unique id is assigned to each service which enables the other components like orchestration and collaboration to interact with the service using the unique id. The "utilized by" attribute keep track of the applications which invoke the service. This helps in keeping the user's profile and maintaining the history of the users utilizing the service. "Utilized at" property means that the service is invoked at time t. This property helps in maintaining the history of the services utilization with respect to time. "Task accomplished" property maintains the history of the service by the tasks it successfully completed; By task completion we means the queries successfully entertained. The "user in consultation" property keeps the record of the service orchestration which enables the orchestration component to build a trust mechanism for the service when an orchestration is need in future. The trust index increases as the number of times successful orchestration is carried out among the services. The "trust index with other services" shows the trust index of the service with other services. " Service consumed by Person ID " property keeps track of the persons who utilizes the service. The property " Service consumed at " keeps track of the location from which the service is invoked.

ServiceID	Name	Utilized by	Utilized at	Task accomplished	Used in consultation	Trust index with other services	Service consumed by Person ID	Service consumed at

3.2.2.9. Entertainment Context Collector

This component is collecting the contextual information of the multimedia devices and contents. The devices play multimedia contents over a period of time with respect to its context and users, therefore, this component play a vital role in understanding the user behavior and enhancing the recommendation performance. The contextual information collected by this component is stored in the EtLR & SSP. This component ensure protecting to expose the services information or the personal information of the users to the SSP. The personal information is collected by this component which is only stored in the EtLR and not in SSP which ensure to preserve the privacy of the users. The following data is collected and stored by this component.

3.2.2.10. Context based Movies Recommendation Notifications

This component generate the notifications based on the recommended movie in a specific context. It also considers the user preferences in order to provide notifications. Both the push and pull notification mechanism is adopted for the ease of the users. The pull notification requires an explicit request from the user while the push notifications need the user's frequency preferences. Based on the user's set frequency, this module sends the notifications to the preferred device.

3.2.2.11. Entertainment Services Orchestration (EtSO)

Service orchestration is the process of combining multiple service to perform a task which cannot be handled by the atomic services. Therefore, a group of services are selected by the service selection module for orchestration and an orchestrated service is created for the complex tasks. The service selection module selects the services from the pool of registered services which in this case are: Accelerometer service, Gyroscope service, RFID service, Location

service, Devices control service, Apriori service, Deep learning service, Context based recommendation service, and Performance evaluation service. The orchestration is carried out from these services which enables to perform a complex tasks such as annotation of the user activities and notification of the activities in smart home. Multimedia content recommendation service is the orchestration of location service, RFID service, Apriori service, and notification service.

$$Recommendation(s) = orchestration\begin{cases} Location \\ RFID \\ Apriori \\ Notification \end{cases} \quad (3.27)$$

3.2.2.12. Entertainment Local Repository (EtLP)

This repository keeps the local services information about the activities in smart office. The following table is maintained in this repository:

ServiceID	Name	Utilized by	Utilized at	Task accomplished	Used in consultation	Trust index with other services	Service consumed by Person ID	Service consumed at

The ServiceID is the id of the registered service and Name is the name of the registered service. A unique id is assigned to each service which enables the other components like orchestration and collaboration to interact with the service using the unique id. The "utilized by" attribute keep track of the applications which invoke the service. This helps in keeping the user's profile and maintaining the history of the users utilizing the service. "Utilized at" property means that the service is invoked at time t. This property helps in maintaining the history of the services utilization with respect to time. "Task accomplished" property maintains the history of the

service by the tasks it successfully completed; By task completion we means the queries successfully entertained. The "user in consultation" property keeps the record of the service orchestration which enables the orchestration component to build a trust mechanism for the service when an orchestration is need in future. The trust index increases as the number of times successful orchestration is carried out among the services. The "trust index with other services" shows the trust index of the service with other services. " Service consumed by Person ID " property keeps track of the persons who utilizes the service. The property " Service consumed at " keeps track of the location from which the service is invoked.

3.3. Processing layer

Processing layer of the architecture handles all the core data processing i.e. data analysis, recognition, classification, prediction through machine learning and statistical approaches. The flow model of the techniques deployed in processing layer is illustrated Figure 3.7. The processing layer performs its functionality in two modes i.e. online and offline. In an online mode the models of the processing layer are trained incrementally with the arrival of the new instances. However, online training is not well suited for the time critical applications. Therefore, another approach which learn/train the model in offline mode meaning that the model collect data over time and update the training model when sufficient data is collected and the system is not required to process the new queries. The flow of the models work in the following fashion. The deployment of predictive machine learning techniques are based on the requirements of the applications. The core concept of how these modules interacts with each other and how the process is carried out is depicted in Figure 3.7 without mentioning the technique used.

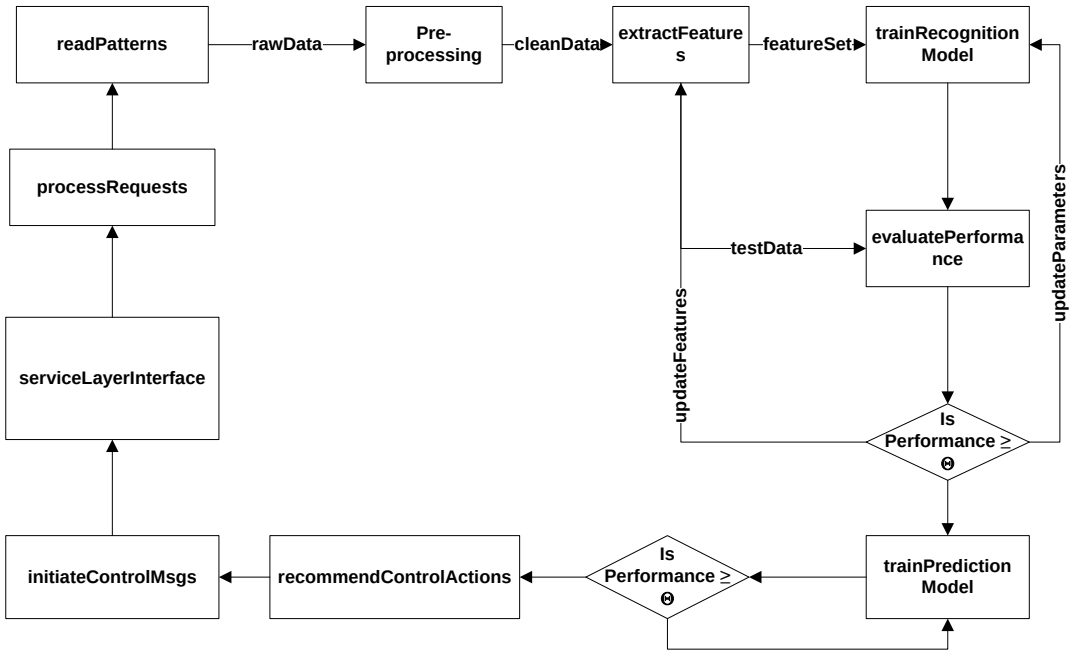


Figure 3.7: Processing layer flow model

The predictive unit enhances the performance of the applications with the state of the art machine learning techniques. The predictive unit predict the context for the recommendation unit which help in preparing proactive response for the situation. The following machine learning techniques are implemented at this layer:

- Collaborative context prediction using predictive rule mining
- Neural network
- Genetic programming
- Regression Tree
- Apriori
- kNN classifier
- Bayesian based Deep Learning Model (BDLM)

The functionality of this layer varies from application to application depending on the requirements of the application, therefore, to avoid redundancy, the detail of the processing

procedure is limited to the experimental chapter, however, a short introduction to each module is presented in the following section.

3.3.1. Common Processing Components

Like the service layer this layer is also divided into two sections i.e. common processing components and application specific processing components. The common processing components are used for pre-processing which clean the raw data. The application specific processing components are used to process the data. Detail of both the architectural section and their components is given below.

3.3.1.1. Outlier Removal (OR)

Outlier removal is the process of removing the unexpected data from the signal received from the sensors. For example, a temperature value cannot be more than a certain threshold value, however due to some error i.e. communication error or device error, it produces a value above the defined threshold, then the system should handle this situation. Different outlier removal techniques have been proposed in the literature, however, we used the window based averaging mechanism to handle the outlier removal. We store the most recent 10 values in the temporary buffer from the sensors and when an outlier is detected by the threshold mechanism then the value is replace through an average of the recent buffer values.

3.3.1.2. Handling Missing Values (HMV)

Missing values are one of the major cause in the performance reduction of the machine learning techniques. Many approaches have been devised to handle missing values. In this work, we used the window based averaging mechanism to handle the outlier removal. We store the most recent 10 values in the temporary buffer from the sensors and when an outlier is detected

by the threshold mechanism then the value is replace through an average of the recent buffer values.

3.3.1.3. Normalization

This is one of the key step in data transformation to same scale. When the parameter values are of different scale then normalization becomes a necessity. Therefore, all parameters should have the same scale for a fair comparison between them. Normalization scales all numeric variables in the range [0,1]. One possible formula is given below:

The formula used for normalization in this work is given below:

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{3.28}$$

3.3.1.4. Data Fusion

The process of collecting data from multiple sources is considered as data fusion. In this work, data fusion component collects data from multiple sources/sensors and forward it to the machine learning models for further processing. The data fusion component ensures that no data attribute left empty and hence the information flow (inputs) to the machine learning model is consistent.

3.3.1.5. Aggregation

The process of aggregating the data from multiple sources and packaging it as a unit data is considered as aggregation. This component is mainly used by the energy consumption experiment where data from multiple apartments are aggregated at a higher spatial granularity level i.e. floor level and building level.

3.3.1.6. Feature Extraction

One of the prominent factor on the performance of the machine learning models is the extraction of best features set from the data. When the input data to an algorithm is too large to be processed and it is suspected to be redundant (e.g. the same measurement in both feet and meters, or the repetitiveness of images presented as pixels), then it can be transformed into a reduced set of features (also named a "features vector"). This process is called feature extraction. The extracted features are expected to contain the relevant information from the input data, so that the desired task can be performed by using this reduced representation instead of the complete initial data. In this work, the process is started with smallest number of features and grow with the performance of the model. The detail on the number and type of features is given in the respective experimental section, however, for activity recognition, we extract features in both time and frequency domains.

Time-domain features contain the basic statistics of each data segment and those of different segments.

- Mean. The mean value of each segment in each dimension.
- Max, Min. The maximum and minimum values of each segment in each dimension. Standard deviation,
- Variance. The variance (and standard deviation) of each segment.
- Correlation. Correlation is calculated between each pair of axes of the acceleration data.
- Signal-Magnitude Area (SMA). SMA is calculated as the sum of the magnitude of the three axes acceleration within the segment window[23]. Besides SMA, there exist similar features to combine the three axes readings.

- Average Resultant Acceleration is the average of the square root of the sum of the values of each axis.
- Another similar feature is the deviation of the sum of the square of acceleration along three axes[33].
- The square root of the sum of the acceleration is used as the movement intensity feature in Ref. [30].

Frequency-domain features describe the periodicity of the signal, which are typically calculated based on the FFT.

- Energy. The energy feature is calculated as the sum of the squared discrete FFT component magnitudes. Ravi et al.[19] used the normalized energy divided by the window length.
- Entropy. The entropy feature is calculated as the normalized information entropy of the discrete FFT components, and it helps in discriminating the activities with the similar energy features[16] .
- Time between peak. This feature is the time between the peaks in the sinusoidal waves[32] .
- Binned distribution. This feature is essentially the histogram of the FFT and it is calculated as follows[32]. First, determine the range of values for each axis (e.g., maximum and minimum). Then, divide this range into 10 equal sized bins, and calculate the fraction of the values falling within each of the bins.

3.3.1.7. Pre-Process Selector (PPS)

This module deals with the selection of pre-processor for the data. Since the number of pre-processor modules are integrated, therefore this module selects the pre-processing module based on the type of data and nature of processing it can provide.

3.3.1.8. Data Smoothening (DS)

The process of smoothening of the sensor signal is called as data smoothening. A smoother fits a smooth curve through data, the fit is called the smooth (the signal), the residuals are called the rough:

Data = Smooth + Rough

- The Signal is assumed to vary smoothly most of the time, perhaps with a few abrupt shifts.

The Rough can be considered as the sum of

- additive noise from a symmetric distribution with mean zero and a certain variance, and
- impulsive, spiky noise: outliers.

In this work, we have used the following

- Moving average
- Loess
- Lowess
- RLoess
- RLowess
- Savitzky-Golay

3.3.2. Application Specific Processing Components

3.3.2.1. Activities Specific Processing Components

First Order Inductive Learner (FOIL): FOIL is proposed by Ross Quinlan in 1993 [58], is a greedy algorithm that learns rules which differentiate between the positive and negative examples. FOIL continues removing the positive examples which are covered by the current best rule until all the positive examples are removed. As a multi-class problem (multiple feature are to be predicted) is considered for evaluation of the proposed system, therefore FOIL is applied on each class: during each run the examples of that class are considered as positive examples and those of other classes are considered negative examples. After the FOIL finishes execution for all the classes the rules are merged to form a uniform rules set. Foil Gain is calculated for each instance before adding it to the current rule. Let suppose a rule R and there are |X| positive examples and |Y| negative examples satisfying the R's body. After adding the instance x to R, there are |X*| positive and |Y∗| negative examples satisfying the new Rule, R*'s body. The Foil gain of x is defined as;

$$gain(x) = \left|X^*\right| \left[\log \frac{\left|x^*\right|}{\left|x^*\right| + \left|y^*\right|} - \log \frac{\left|x\right|}{\left|x\right| + \left|y\right|} \right] \tag{3.30}$$

Where gain(x) is the number of bits saved in representing all the positive examples by adding x to R.

FOIL ALGORITHM

INPUT:

Training set S = X ∪ Y. (X and Y are the sets positive and negative instances, respectively.)

OUTPUT:

Rule set R that predict class label of all instances.

Procedure FOIL

```
R ← Φ        //empty rule set
While |X| > 0
    Y0 ← Y , X0 ← X
    rule r ← empty rule
    While |Y0| > 0 and r.length < max_rule_ length
        Find the instance x with highest gain according to X0 and Y0
        Append x to r
        Remove from X0 //all examples not satisfying r
        Remove from Y0 //all examples not satisfying r
    End
    R ← R ∪ {r}
    Remove from X all examples satisfying r's body
End
End return R
```

Predictive Rule Mining (PRM): Predictive Rule Mining (PRM) is an extension to the FOIL for higher accuracy and efficiency. The FOIL generates very small number of rules due to which it does not achieve higher accuracy. The PRM, assign weight to each rule in the rules set R. PRM decreases the weight at a certain ratio of rule after an instance is successfully covered by the rule. The PRM has greater number of rules as compared to FOIL and each positive instance has more covering rules.

The FOIL consumes most of the time on evaluating each instance during the calculation of highest gain. A technique similar to [59] is used in this study.

SInfo stores the following information corresponding to rule r.

- X and Y: The number of positive and negative instances that satisfy R's body

- X (x) and Y(y): For all possible instances of x, satisfying body of rule r.

The gain will be calculated for each instance using the information in SInfo corresponding to the current rule. The gain calculation is independent of the size of dataset.

PRM ALGORITHM

INPUT:

Training set S = X ∪ Y. (X and Y are the sets positive and negative instances, respectively.)

OUTPUT:

Rule set R that predict class label of all instances.

Procedure PRM

```
Set the weight of every example to 1
Rule Set R ← Φ
t_weight ← t_weight (X)
A ← Compute SInfo from R
While t_weight (X) > α.totalWeight
    X0 ← X, Y0 ← Y, A0 ← A
    Rule r ← empty rule
    While true
        Find best instance x according to A0
        If gain (x) < min_gain then break
        Append x to r
        For each example t in X0 ∪ Y0 ≠ r's body
            Remove t from X0 or Y0
            Change A0 according to the removal of t
        End For
  End While
  R ← R ∪ {r}
  For each example t in X satisfying r's body
     t.weight ← α·t.weight
     change A according to the weight decreased
```

End For

End While

return R

End PRM

The procedure of using the rule for prediction is as follow:

- Select all the rules whose bodies are satisfied by the instances
- Select the best k rules for each class from the rule set of step 1
- Calculate the accuracy for each rule using equation mentioned in FOIL
- Compare the accuracy for each class and choose the class with highest accuracy as the predicted class.
- When the system does not find the pattern in the history of current user, it checks the user category/type. When the system finds other users of the same category/type, it checks their histories for the presented pattern. Upon the positive results from the other histories, the system performs the appropriate action.

The following terminologies are referred very often in this work.

- Profile: all information that relates to a particular entity, e.g., information about the type of a person/user, location, time, duration, participants, etc. The profile can be updated and can evolve over time.

Category: the category of an entity is a set of other similar entities that share some basic characteristics, e.g., similar actions and histories of the same type or similar people with same interests.

k-Nearest Neighbor based annotation(kNN): KNN is an instance-based classifier based on the majority voting of its neighbors [43]. In general, KNN is one of the most popular algorithms for pattern recognition[24] and is par with Decision Tree in terms of performance and the computational complexity. In [44], Lombriser et al. developed a dynamic sensor network combining KNN and decision tree algorithm for activity recognition. According to their test, KNN and J48/C4.5 are identified as "the classifiers with the least complexity but rendering acceptable performance". In [24], Kaghyan and Sarukhanyan adopted KNN in the desktop application of activity recognition using acceleration data recorded by smart phones. They observed that the choice of a good training set is the key factor in the recognition accuracy. Kose et al.[45] developed Clustered KNN for their online activity recognition system. In this work, we applied kNN for annotation of the accelerometer and gyroscope data which helps in identifying the contextual information about the users.

Activities Parameter Selection: Parameter selection plays a central role in machine learning. The main idea of parameter selection is to choose a subset of relevant parameters for building robust learning models. There are many potential benefits of parameter selection: facilitating data visualization and data understanding, reducing the measurement and storage requirements, reducing training and utilization times, defying the curse of dimensionality to improve prediction performance. There are ad hoc parameter selection techniques, but there are also more systematic approaches. From a theoretical perspective, it can be shown that optimal parameter selection requires an exhaustive search of all possible subsets of parameters. If large numbers of parameters are available, this is impractical. For machine learning, the search is for a satisfactory set of parameters instead of an optimal set.

In this work, two machine learning approaches are utilized i.e. kNN and FOIL based PRM, therefore we have adopted an ad hoc parameter selection for both the approaches. The number of

nearest neighbors are and optimal rules set are empirically derived through extensive experimentations on the dataset.

Activities Performance Evaluation: In order to test the system more extensively, data is divided into four sets of training data sets for every testing data set which are used to construct the rules for the context prediction model. The first training data set contains the information of the context histories of one type of users. The test data is generated from the same type of users. In the second training set the context information of the two types of user are used for testing the system. For the third test we extend the context information to three types of users. For the whole testing set the information of all the users were utilized for training. These data sets were formed in order to test the dependability of the types of users on each other. Every time, new context data is added, the rules and prediction model are updated. This information is used to forecast the context of the users which are the part of the test dataset.

The accuracy of rules needs to be evaluated before applying on prediction. As we have a total of 9 parameters for our problem therefore our rule look like:

$$\{a_1 \wedge a_2 \wedge a_3 \wedge a_4 \wedge a_5 \wedge a_6 \wedge a_7 \wedge a_8 \wedge a_9\} \rightarrow \{C_i\}$$

The rules are evaluated using Laplace expected error estimate. The rule's accuracy estimation is given as:

$$LaplaceAccuracy(i) = \frac{x_i+1}{x_t+k} \tag{3.31}$$

Where x_t is the total number of instances covered by the rule, x_i is the positive instances covered and k is the total number of classes.

3.3.2.2. Energy Specific Processing Components

Neural Network (NN): Neural networks are the most widely used artificial intelligence models in the application of building energy prediction. This type of model is good at solving non-linear problems and is an effective approach to this complex application. In the past twenty years, researchers have applied NNs to analyze various types of building energy consumption in a variety of conditions, such as heating/cooling load, electricity consumption, sub-level components operation and optimization, estimation of usage parameters [60][61][62]. The NN has many variants like MLPs, SOM, FFNN, BPNN and RNN, however for the purpose of this work FFNN has been used which has been proven best in load forecasting and commercial building consumption forecasting.

FFNN is elaborated for the day ahead consumption forecast in a residential building. FFNN can learn to approximate a function f that maps Rm → R without making assumptions about the relationship between the input and outputs. While a FFNN does not make assumptions about the inputs or outputs, it does require the user to define the model's structure, including the number of hidden layers and hidden units within the network and any other associated parameters. FFNN with a single hidden layer for function approximation has the following mathematical representation:

$$f(x) = \sum_{j=1}^{N} w_j \psi_j \left[\sum_{i=1}^{M} w_{ij} x_i + w_{io} \right] + w_{jo} \tag{3.32}$$

Where N represents the total number of hidden units, M represents the total number of inputs, and Ψ represents the activation function for each hidden unit.

In this work, the following network structure is used:

- single hidden layer (with varying number of neurons)

- tan-sigmoid(x) as an activation function at the hidden layer
- linear function at the output layer

Levenberg–Marquardt algorithm [63] is used for training, which is one of the most popular training algorithm for FFNNs with gradient decent based method. We observe during the experimentation that, if the training data is short then it leads to a low accuracy in forecasting the 24 hour ahead consumption. Once the model is trained to forecast a single step ahead consumption, our system presents the next input pattern and forecast the consumption. As long as the error between the forecasted value and the actual value remains in the acceptable range, the system continuously forecast the consumption. When the error over pass the acceptable range, the system triggers the training module and retrains the model with the most recent observed patterns.

Classification And Regression Tree (CART): In recent times, there has been increasing interest in the use of CART analysis. CART analysis is a tree-building technique which is different from traditional data analysis methods. In a number of studies, CART has been found to be quite effective for creating decision rules which perform as well or better than rules developed using more traditional methods [64][65]. In addition, CART is often able to uncover complex interactions between predictors which may be difficult or impossible using traditional multivariate techniques.

The methodology of CART used in this work is outlined here. The enhanced recursive partitioning algorithm [66] is utilized for CART. This enhanced recursive partitioning improves the performance of the CART and also improves the convergence time. The algorithm work in a step-by-step process by which a decision tree is constructed by either splitting or not splitting each node on the tree into two child nodes.

The model input are defined as:

$$X(t) = [y(t-30; \ldots; y(t-1); h; d; dow; m; T(t); H(t)]$$

where y(t - 30); ... ; y(t - 1) is the electricity consumption for the last 30 days, h is set of hours, d is the set of days, dow is the day of week, m is month of year, T(t) is set of temperature values, and H(t) is the set of humidity values in the given time period. The 30 days data is used for training due to the reason that the daily pattern exhibits the same behavior which leads to a good forecast.

Genetic Programming (GP): Genetic programming is an evolution means of representing artificial intelligence action by means of programs, and its basic idea is the Theory of Evolution obeying to the natural rules of selecting the superior and eliminating the inferior, the survival of the fittest[67]. Genetic programming inherits the basic idea of Genetic Algorithm (GA), but contrarily it is different from Genetic Algorithm.

Genetic Programming starts with randomly creating an initial search space of individual computer programs composed of the available functions and terminals first, and each individual in the search space has its own fitness.

Fitness of the individual is evaluated by minimizing the value of MAPE using the equation:

$$Fitness\_function = Minimize\left(\frac{1}{N}\sum_{i=0}^{N}\frac{y_i - \bar{y}_i}{y_i}\right) \tag{3.33}$$

As many as functions that reflect the essence of variables and load value of each time point are selected, for the relationship between them cannot be determined well and truly beforehand. Here the functions are F = {+, −, *, /, sin, cos, sec, tan, ln, exp, sqrt} and the terminals are T = {h, d, dow, m, ρ}, where h is the hour, d is the day, dow is the day of week, m is the month, and ρ is the electricity consumption.

Incremental learning of NN (Self training): Self-training is a commonly used method for semi-supervised learning. In this method a predictor is first trained with the sample of training data. Then the trained classifier is used to classify the unlabeled datasets. Normally the most assured unlabeled data, along with their predicted labels, are appended to the training set. The classifier is re-trained with the new data and the process is repeated. This process of re-training the classifier by itself is called self-teaching or bootstrapping.

The problem of multi-step electricity consumption forecasting (long-term forecasting) is an ideal problem to be solved with the incremental learning. The reason is that the data collected is for one year while we want to extend the forecast for a longer period, so the available consumption data for training is shorter than the forecasting data. In this work, the data is recorded for 1 year i.e. 2010 and a long term forecasting is carried out for the ease of energy management companies. Therefore, a novel incremental model is devised to forecast a long term forecasting using a small amount of training data. This is achieved with the below procedure:

The train model continues to forecast the step ahead consumption and evaluate the error difference between the predicted consumption and actual consumption. The threshold value is set by the user to tell the model the acceptable error of the model. Once the model's error crosses the acceptable range, the system retrains the model as shown in Figure 5.6.

Ensemble Model (Co-training): An ensemble approach based on co-training semi-supervised learning technique is proposed for the enhanced performance of the electricity consumption forecasting. co-training has two *views* of the data. It assumes that each example is described using two different feature sets that provide different, complementary information about the instance. Ideally, the two views are conditionally independent (i.e., the two feature sets of each instance are conditionally independent given the class) and each view is sufficient (i.e., the class of an instance can be accurately predicted from each view alone). Co-training first

learns a separate regressor for each view using any training data. The most confident predictions of each regressor on the testing data are then used to iteratively construct additional training data.

A co-training style semi-supervised regression algorithm, i.e. COREG, is proposed in [68]. This procedure uses two regressors where one regressor labels the unlabeled data for the other regressor. The COREG style of co-training mechanism is adopted for carrying out experiments, however the based regressors are different. We used the supervised learning techniques (GP, RTREE, and FFNN) as the base prediction techniques. The hybrid model is developed from the supervised techniques. This supervised model is then extended for unlabeled data as in a co-training mechanism.

Energy Parameter Selection: The model inputs are considered in consistence with the work presented in the [69][70] which are defined as:

$$X(t) = [y(t-1); y(t-2); T(t); s; sh; ch]$$

where y(t - 1) and y(t - 2) represent known electrical consumption values for the previous two time steps, T(t) is the current temperature, S (t) is the current solar flux, s is an indicator variable that denotes weekend/holiday or weekday, sh is the sine of the current hour and ch is the cosine of the current hour. Due to non-availability of data presented in the equation 4, we define the model input for testing on our real data set as:

$$X(t) = [y(t-30; ....; y(t-1); h; d; dow; m; T(t); H(t)]$$

where y(t - 30); ... ; y(t - 1) is the electricity consumption for the last 30 days, h is set of hours, d is the set of days, dow is the day of week, m is month of year, T(t) is set of temperature values, and H(t) is the set of humidity values in the given time period. The 30 days data is used

for training due to the reason that the daily pattern exhibits the same behavior which leads to a good forecast.

Energy Performance Evaluation: We evaluate the performance of our model with Mean Absolute Percentage Error (MAPE) which is consistent with [4][6].

$$MAPE = \frac{1}{N}\sum_{i=0}^{N}\frac{y_i - \bar{y}_i}{y_i} \tag{3.34}$$

where y_i is the actual consumption and $\bar{y}_i$ is the predicted value and N is the total number of consumption hours.

3.3.2.3. Entertainment Specific Processing Components

Collaborative Filtering using Apriori (ARM): The goal of recommendation systems is to suggest items to a particular user. A user and an item are the basic entities that appear in a recommendation system. A user is a person who utilizes the recommender system providing her/ his opinion (i.e., rating) about various items. Then, a user receives recommendations about new items from the system based on her/his opinion [2]. The task of recommendation systems is to predict the ratings of the items that the user has not seen or ranked before [3]. Based on that predicated rating, the recommendation system will be able to recommend other items to the user [3]. There are different approaches to recommendation systems that are used to serve in different contexts based on system needs [4]. Our work is to combine association rules mining and content-based approach to provide a framework of a hybrid recommendation system on two dimensional spaces (*User* × *Item*).

Our approach is divided into the following steps:

• The first step is to apply the Apriori algorithm to a (*User* × *Item*) matrix to generate the association rules.

• The second step is to divide the items that have been rated by users into two categories: *Favorite Items* and *Non-Favorite Items*.

• The third step is to use the generated association rules to discover the frequent itemsets of *Favorite Items* set and find the correlations among those items to recommend new items to a target user.

• The last step is to apply the content based approach into *Non-Favorite Items* set to recommend some new items to a target user.

The Apriori algorithm is a well-known algorithm that is used for mining frequent itemsets for Boolean association rules [5]. It is an algorithm for efficient association rule discovery [6]. The algorithm was proposed by R. Agrawal and R. Srikant in 1994 [5]. The approach that is used in the the Apriori algorithm is known as a level-wise search, where k-itemsets are used to explore (k+1)-itemsets [5]. This study introduced an algorithm that decrease the number of candidate items in the candidate item set Ck . In the Apriori algorithm, Ck-1 is compared with support level once it was found. Item sets less than the support level will be pruned and Lk-1 will come out which will connect with itself and lead to Ck . The optimized algorithm is that, before the candidate item sets Ck come out, further prune Lk-1, count the times of all items occurred in Lk-1, delete item sets with this number less than k-1 in Lk-1. In this way, the number of connecting items sets will decrease, so that the number of candidate items will decline.

The following is the description of the optimized algorithm:

Input: content DB, devices DB, user History

D: minimum support level threshold is minsup

Output: frequent item sets L in D

1) L1=frequent_1-itemsets(D);

2) For (k=2;Lk-1≠φ;k++);

3) Prune1(Lk-1);

4) Ck=apriori_gen(Lk-1;minsup);

5) for all transactions t∈ D {

6) C= sumset (Ck ,t); find out the subset including Ck

7) for all candidates c∈ Ct

8) { c.count ++; }

9) Lk ={c∈ Ck |c.count≥minsup} //result of Prune 2(Ck) } }

10) Return Answer∪ kLk

Algorithm: Prune Function:

Input: set k-1 frequent items of Lk-1 as input parameter

Output: go back and delete item sets with this number less than k-1 in Lk-1

Procedure

Prune 1(Lk-1) 1) for all itemsets L1∈ Lk-1 2)

if count(L1)≤k-1 3) then

delete all Lj from Lk-1 (L1∈ Lk-1) 4)

reture L'k-1 // go back and delete item sets with this number less than k-1 in Lk-1

Bayesian based Deep Learning Model: Similar to the work presented in [71], the recommendation task considered in this work takes implicit feedback [72] as the training and test data. The entire collection of J items (movies) is represented by a J-by-S matrix X_c, where row j is the bag-of-words vector $X_{c,j*}$ for item j based on a vocabulary of size S. With I users, we define an I-by-J binary rating matrix $R = [R_{ij}]_{I \times J}$. For example, in the dataset "movie" & "netflex" $R_{ij} = 1$ if user i has article j in his or her personal library and $R_{ij} = 0$ otherwise. Given part of the ratings in R and the content information X_c, the problem is to predict the other ratings in R. Although we focus on movie recommendation (where plots of movies are considered as content information) our model is general enough to handle other recommendation tasks (e.g., tag recommendation, article recommendation).

The matrix X_c plays the role of clean input to the Stack Denoising Auto Encoders (SDAE) while the noise-corrupted matrix, also a J-by-S matrix, is denoted by X_0. The output of layer l of the SDAE is denoted by X_l which is a J-by-K_l matrix. Similar to X_c, row j of X_l is denoted by $X_{l,j*}$. W_l and b_l are the weight matrix and bias vector, respectively, of layer l, $W_{l,*n}$ denotes column n of W_l, and L is the number of layers. For convenience, we use W^+ to denote the collection of all layers of weight matrices and biases. Note that an L/2-layer SDAE corresponds to an L-layer network.

We first briefly review SDAE and give a Bayesian formulation of SDAE. This is then followed by the presentation of CDL as a hierarchical Bayesian model which tightly integrates the ratings and content information.

Stacked Denoising Autoencoder: SDAE [73] is a feed forward neural network for learning representations (encoding) of the input data by learning to predict the clean input itself in the output. Usually the hidden layer in the middle, i.e., X2 in the figure, is constrained to be a

bottleneck and the input layer X0 is a corrupted version of the clean input data. An SDAE solves the following optimization problem:

$$\min_{\{W_j\},\{b_j\}} \|X_c - X_L\|_F^2 + \lambda \sum_l \|W_l\|_F^2 \tag{3.35}$$

where λ is a regularization parameter and $\|.\|_F$ denotes the Frobenius norm.

Bayesian SDAE: If we assume that both the clean input X_c and the corrupted input X_0 are observed, similar to [74]–[77], we can define the following generative process:

- ✓ For each layer *l* of the SDAE network,
- For each column n of the weight matrix W_l, draw

  $W_{l,*n} \sim \mathrm{N}(0, \lambda_w^{-1} I_{Kl})$.
- Draw the bias vector $b_l \sim N(0, \lambda_w^{-1} I_{Kl})$.
- For each row j of X_l, draw

  $X_{c,j*} \sim N\left(\sigma\left(X_{l-1,j*}, W_l + b_l\right), \lambda_s^{-1} I K_l\right)$.
- ✓ For each item j, draw clean input

$$X_{c,j*} \sim N\left(X_{L,j*}, \lambda_n^{-1} I_J\right).$$

Deep Learning Model (DLM): We followed the model presented in [78] and Bayesian SDAE as our base model for the content recommendation task. The steps followed are as follows:

1. For each layer l of the SDAE network,

   (a) For each column n of the weight matrix W_l, draw

$$W_{l,*n} \sim \mathrm{N}(0, \lambda_w^{-1} I_{Kl})$$

   (b) Draw the bias vector $b_l \sim N(0, \lambda_w^{-1} I_{Kl})$.

(c) For each row j of X_l, draw

$$X_{l,j*} \sim N\left(\sigma\left(X_{l-1,j*}, W_l + b_l\right), \lambda_s^{-1} I K_l\right).$$

2. For each item j,

(a) Draw a clean input $X_{c,j*} \sim N\left(X_{L,j*}, \lambda_n^{-1} I_J\right)$.

(b) Draw a latent item offset vector $\epsilon_j \sim N(0, \lambda_v^{-1} I_K)$ and then set the latent item vector to be:

$$v_j = \epsilon_j + X_{\frac{L}{2},j*}^T$$

3. Draw a latent user vector for each user *i*:

$$u_i \sim N(0, \lambda_u^{-1} I_K)$$

4. Draw a rating R_{ij} for each user-item pair (i, j):

$$R_{ij} \sim N(u_i^T v_j, C_{ij}^{-1})$$

Here λ_w, λ_n, λ_u, λ_s, λ_v are hyper-parameters and C_{ij} is a confidence parameter. The middle layer $X_{L/2}$ serves as a bridge between the ratings and content information. This middle layer, along with the latent offset ϵ_j, is the key that enables DLM to simultaneously learn an effective feature representation and capture the similarity and (implicit) relationship between items (and users). Similar to the generalized SDAE, for computational efficiency, we can also take λ_s to infinity.

Prediction with DLM: Let D be the observed test data. Similar to [34], we use the point estimates of u_i, W^+ and ϵ_j to calculate the predicted rating:

$$R_{ij}^{*} \approx (u_j^{*})^T (f_e\left(X_{0,j*}, W^{+*}\right)^T + \epsilon_j^{*}) = (u_i^{*})^T v_j^{*} \tag{3.36}$$

Note that for any new item *j* with no rating in the training data, its offset ϵ_j will be 0.

Entertainment Performance Evaluation: In order to measure the closeness of predictions to users' real preferences, a numerical representation is normally used. In addition, for reasons of clarity, we will use the same notation along the current section. To this end, let us call P(u, i) the predictions of a recommender system for every particular user u and item i, and p(u, i) the real preferences. Clearly, the function p(u, i) can never be known with absolute precision. Therefore, the values of this function are most usually estimated by means of the users' previous ratings. As we said above, these ratings can be obtained explicitly or implicitly. In some cases, both functions p(u, i) and P(u, i) will offer only two values 1 or 0, which means that a particular item I is considered useful or useless, respectively, for a particular user u. For this singular case, we will say that p and P are binary functions.

Accuracy metrics measure the quality of nearness to the truth or the true value achieved by a system. Perhaps, accuracy is the most used and well-known metric into the field of Artificial Intelligence. Particularizing the metric to the recommender system's field, it can be formulated as shown:

$$accuracy = \frac{number\ \ of\ \ good\ \ cases}{number\ \ of\ \ cases} \tag{3.37}$$

$$accuracy = \frac{number\ \ of\ \ successful\ \ recommendation}{number\ \ of\ \ recommendations} \tag{3.38}$$

Now, assuming that a ''successful recommendation'' is equivalent to ''the usefulness of the recommended object is close to the user's real preferences'', and using the functions *p* and *P* introduced previously, we may be more formal and reformulate accuracy as in (5). In this equation, we consider that *p* and *P* are binary functions. Additionally, *r(u, i)* is 1 if the

recommender showed the item i to the user u, and 0 otherwise. Finally$R = \sum_{u,i} r(u,i)$is the number ofrecommended items shown to the users.

$$accuracy = \frac{\sum_{(\forall u,i/r(u,i)=1)} 1-|p(u,i)-P(u,i)|}{R} \quad (3.39)$$

Also common in the recommender systems' field is the metric mean absolute error (MAE). This metric measures the average absolute deviation between each predicted rating *P(u, i)* and each user's real ones *p(u, i).* Then, due to the fact that only rated items can show us each user's real preferences, we may derive the (3.40), where i must have been rated by u (to obtain p(u, i)). In this, we consider N as the number of observations available, which obviously depends on the number of items properly rated. Of course, the higher this number, the better the estimate.

$$MAE = \frac{\sum_{u,i}|p(u,i)-P(u,i)|}{N} \quad (3.40)$$

3.4. Acquisition & control layer

Data collection and consumption layer is a physical layer in our model, which sensors and actuators are posted to sense the real world and adapt the environment to the user's need through actuators. The sensors data is collected through various sensors, like environmental sensors, motion sensors, position sensors, identification sensors etc. The data collected through sensors are used by machine learning techniques for further processing and recommend a feasible environment to the user. This task is carried out through the actuators. The actuator are used to adapt the environment to the situation like making coffee when a smart home user is getting ready for office.

4. Experiment 1: Collaborative context prediction in smart office

Context prediction plays a vital role in an assistive ubiquitous environment. The environmental configuration in a ubiquitous environment is heavily dependent on the context of the events occurring in the environment. Due to ubiquitous nature of the IoT applications, context prediction can significantly enhance the capabilities of the IoT applications. There has been a dominating applications considering context as a prime source of information for enhancing the performance of the applications [2]. However, the current state of the art context aware techniques brought the inherent problem of the existing solution i.e. the current state of the art approaches utilize the user's history information for predicting the context of the events [79]–[81]. When the user's history does not provide apposite contextual information for the observed activity/event at time *t*, the history based state of the art context prediction techniques fails to predict the appropriate future context [82].

The contribution of this work is twofold i.e. enhancing the performance of context prediction with collaboration and incorporating a case study of collaborative context prediction i.e. activity prediction in smart office into proposed CIoT architecture. The performance of context prediction is enhanced with the Profile based Collaborative Context Prediction (PCCP) approach. PCCP is a predictive association rules based system which utilizes the history of similar users and collaborate among users of the ubiquitous environment. PCCP generates rules at high level of abstraction, human readable and understandable that helps in avoiding the underline details. The layered architecture of the proposed context prediction approach is shown in Figure 4.1. The architecture consists of 4 layers i.e. application layer, service layer, processing layer, and acquisition & control layer.

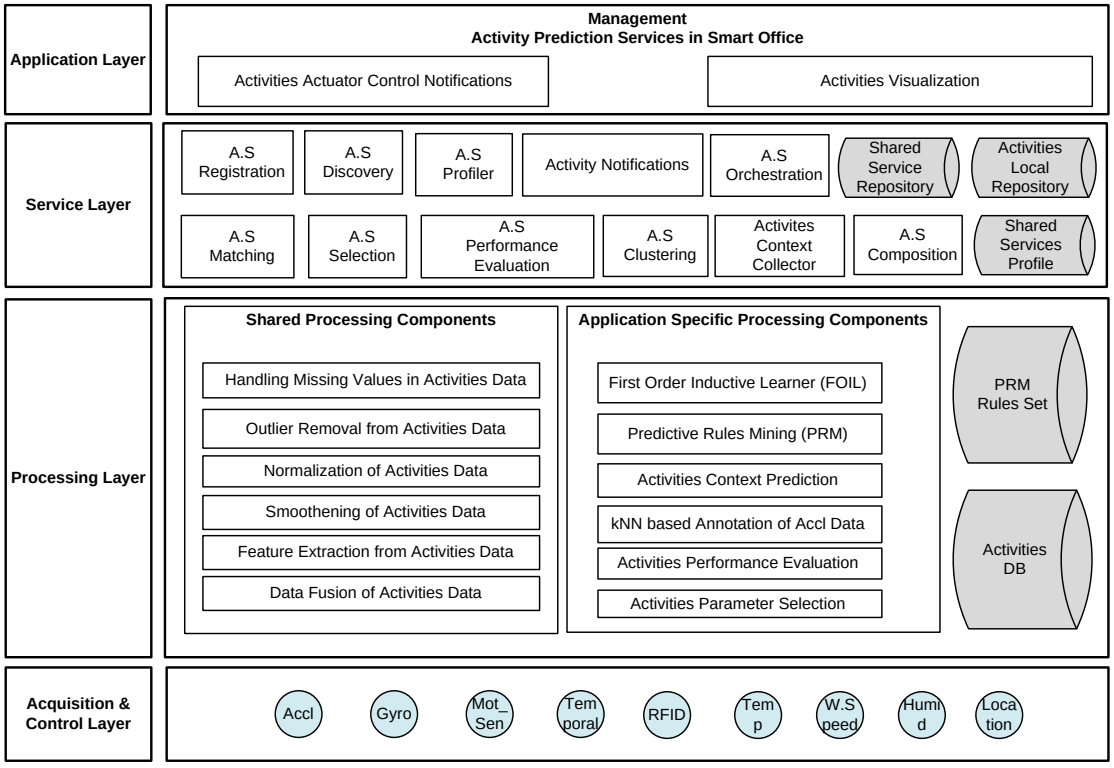


Figure 4.1: Conceptual component diagram of the activity prediction in smart office

Application Layer: This layer is the consumer layer of the processing carried out at the lower layers. The application layer carry out all the interaction with the users, handle the environment level details and proactively adapt the environment according to the user's context and the system's recommendation. As presented in Figure 4.1, two type of output is provided by this application i.e. actuator control notification and activities visualizations. The actuator control notifications are in the form of text messages while the visualization is in the form of interleaved activities with respect to time.

Service Layer: Service layer provides the core functionalities of services registration, orchestration, composition and collaboration. The functionality of the components shown in The components of Figure 4.1 is shortly described in chapter 3, however, the detail of the components utilized for this experiment is given below:

The deployment of this experiment into the CIoT is initiated by the definition of the services. The services are defined for the sensors, actuator and all the processes. As shown in Figure 4.1, Physical sensors like, accelerometer, gyroscope, motion sensors, temperature, humidity and RFID and logical sensors like temporal, location and wind speed are utilized for this experiment. These sensors are considered as "things" which is further considered as a service in CIoT by utilizing the concept of TaaS as mentioned before. Besides, the processing components like Notifications, FOIL, PRM, kNN, parameter selection, performance evaluation are also considered as a service utilizing the concept of PaaS as mentioned before. Collaboration component collaborate among the users of the system which enhances the performance of the activities prediction. The TaaS services are used to control the devices i.e. sensors and actuators while the PaaS services are used to process the data and produce the control notifications. The services are registered with their associated resources and are discovered with the service discovery mechanism.

Processing layer: The central processing layer of the system is processing layer. At this layer, all the key components are involved in processing the collected data. This layer comprised of context aggregation, handling missing values, normalize data and most importantly anonymize data. The widow based averaging as discussed in chapter 3 is used for handling missing values and outliers. The data is stored in the activities database after data collection at the data fusion module. The rules generator generates the rules from the data stored in the activities database using the First order inductive learning (FOIL). The rules are stored in the PRM rules set which is used for context recognition and prediction. The performance evaluator of the system continuously evaluate the performance of the system and modify the rules set accordingly. The context prediction collaborate with the service layer for the processed data.

The rest of the system work as follows: The collaborative filtering concept used in recommendation system as discussed in [83] is incorporated in this work. The primary focus of collaborative filtering is the prediction of a user's interest by utilizing the information from many users' histories. However, the concept is generalized to predict the behavior of an entity (human or object) by utilizing the behavior patterns of many other but similar entities. The similarity is explored between situations and infer the relationship and dependencies. Each situation is a combination of actions. For example an activity a_i has been carried out once at location l_i by person p_i, suppose that the same activity a_i is performed by another person p_j at the same location l_i; this situation shows that there is a relationship between p_i, p_j, a_i, and l_i . This relationship implies that if p_i visits the same location l_i again then the probability $P(a_i)$ will more likely to be ≥ 0. Similarly, if p_i performs a_i in l_j, then p_i, p_j, a_i, l_i and l_j are related, which means that if p_j visits l_i, the probability $P(a_i, p_j)$ will more likely to be ≥ 0. The data collected for experiments will be used to infer the relations.

- a_i is performed by p_i at l_i
- a_j is performed by p_i in l_j
- ……
- a_n is performed by p_i in l_n

This data creates possible relations between $a_i \ldots a_n$, p_i, and $l_i \ldots l_n$, which later can be used for adaptation in similar situations involving same places, people or activities. It has been found that the pattern of *n-m-k* namely with *m* persons involving in *k* activities distributed in *n* locations is the general pattern and all other patterns can be derived from that. Therefore, concentration has been kept on situation scenarios containing several users doing multiple activities spread in different locations. The produced model for this complex situation can be applied for other pattern classes as well.

A smart office is considered as a testing scenario for this work, where multiple persons are involved. Therefore the problem becomes multiple users' action recognition. When it comes to multiple users, feature modeling becomes complex and important. The data is divided into three sets: (1) set of users *P* of the collaborative ubiquitous environment, (2) set of possible context patterns *CP* and (3) set of predictable contexts *C*. This problem is formulated with the sequential association rules as shown in equation 3.1.

$$\langle \{p_i \in P\} \{cp_i \in CP\} \rangle \Rightarrow \{c_i \in C\}$$

The rule can be read as: if for the user $\{pi \in P\}$ the pattern $\{cp_i \in CP\}$ occurs, it implies that $\{c_i \in C\}$ will appear in the future and hence the relevant actions will be performed.

For the purpose of the contexts prediction, Three types of contextual data is considered i.e. Personal, Temporal and Contextual (Environmental, Location). The parameters being considered are;

- User ID
- User Name
- User Category (Profile based category)
- Time
- Location
- Movement Direction
- Temperature
- Humidity
- Wind Speed.

The feature attributes in the pattern set are combinations of persons, pattern set, and actions. This will imply to predict the future context of a person pi in time t_i at location l_i.

Figure 4.2 shows overlapped actions by multiple persons in the same time slot. For example action a_1 is performed by person p_1 at time t_0-to-t_2; however the same action a1 is performed by person p_4 between t_2 and t_7. These kind of overlapping actions are handled in our proposed approach by considering user profiles as central point for collaboration of information.

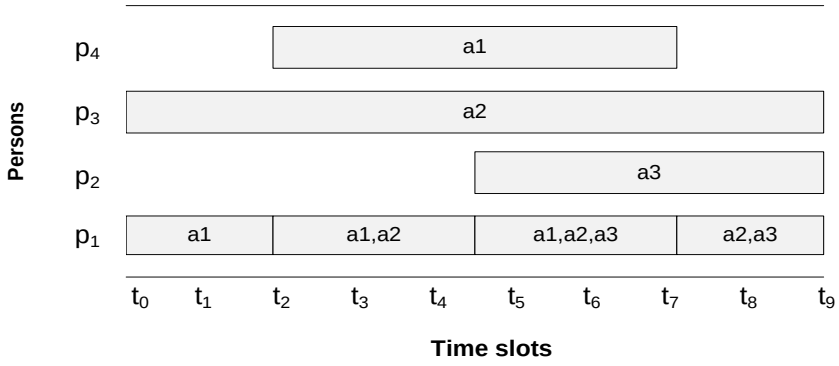


Figure 4.2: Illustration of overlapped activities

Association rules mining [84] is used to extract minimum number of rules for predicting the user's context in a time series. Firstly we generate rule using inductive logic and then apply a predictive rules mining algorithm to enhance the efficiency of rules generated through inductive logic. The best rule set is then applied to find the user's context.

First Order Inductive Learner (FOIL): FOIL is proposed by Ross Quinlan in 1993 [58], is a greedy algorithm that learns rules which differentiate between the positive and negative examples. FOIL continues removing the positive examples which are covered by the current best rule until all the positive examples are removed. As a multi-class problem (multiple feature are to be predicted) is considered for evaluation of the proposed system, therefore FOIL is applied on each class: during each run the examples of that class are considered as positive examples and

those of other classes are considered negative examples. After the FOIL finishes execution for all the classes the rules are merged to form a uniform rules set.

Foil Gain is calculated for each instance before adding it to the current rule. Let suppose a rule R and there are |X| positive examples and |Y| negative examples satisfying the R's body. After adding the instance x to R, there are |X*| positive and |Y∗| negative examples satisfying the new Rule, R*'s body. The Foil gain of x is defined as;

$$gain(x) = |X^*| \left[\log \frac{|x^*|}{|x^*| + |y^*|} - \log \frac{|x|}{|x| + |y|} \right]$$

Where gain(x) is the number of bits saved in representing all the positive examples by adding x to R.

FOIL ALGORITHM

INPUT:

Training set S = X ∪ Y. (X and Y are the sets positive and negative instances, respectively.)

OUTPUT:

Rule set R that predict class label of all instances.

Procedure FOIL

```
R ← Φ        //empty rule set
While |X| > 0
    Y0 ← Y , X0 ← X
     rule  r ← empty  rule
    While |Y0| > 0 and r.length < max_rule_ length
          Find the instance x with highest gain according to X0 and Y0
          Append x to r
          Remove from X0 //all examples not satisfying r
          Remove from Y0 //all examples not satisfying r
    End
    R ← R ∪ {r}
```

Remove from X all examples satisfying r's body

End

End return R

Predictive Rule Mining (PRM): Predictive Rule Mining (PRM) is an extension to the FOIL for higher accuracy and efficiency. The FOIL generates very small number of rules due to which it does not achieve higher accuracy. The PRM, assign weight to each rule in the rules set R. PRM decreases the weight at a certain ratio of rule after an instance is successfully covered by the rule. The PRM has greater number of rules as compared to FOIL and each positive instance has more covering rules.

The FOIL consumes most of the time on evaluating each instance during the calculation of highest gain. A technique similar to [59] is used in this study.

SInfo stores the following information corresponding to rule r.

- X and Y: The number of positive and negative instances that satisfy R's body
- X (x) and Y(y): For all possible instances of x, satisfying body of rule r.

The gain will be calculated for each instance using the information in SInfo corresponding to the current rule. The gain calculation is independent of the size of dataset.

PRM ALGORITHM

INPUT:

Training set S = X ∪ Y. (X and Y are the sets positive and negative instances, respectively.)

OUTPUT:

Rule set R that predict class label of all instances.

Procedure PRM

Set the weight of every example to 1

Rule Set R ← Φ

```
    t_weight ← t_weight (X)
    A ← Compute SInfo from R
    While t_weight (X) > α.totalWeight
        X0 ← X, Y0 ← Y, A0 ← A
        Rule r ← empty rule
        While true
            Find best instance x according to A0
            If gain (x) < min_gain then break
            Append x to r
            For each example t in X0 ∪ Y0 ≠ r's body
                Remove t from X0 or Y0
                Change A0 according to the removal of t
            End For
        End While
        R ← R ∪ {r}
        For each example t in X satisfying r's body
            t.weight ← α·t.weight
            change A according to the weight decreased
        End For
    End While
    return R
End PRM
```

The procedure of using the rule for prediction is as follow:

- Select all the rules whose bodies are satisfied by the instances
- Select the best k rules for each class from the rule set of step 1
- Calculate the accuracy for each rule using equation mentioned in FOIL
- Compare the accuracy for each class and choose the class with highest accuracy as the predicted class.

- When the system does not find the pattern in the history of current user, it checks the user category/type. When the system finds other users of the same category/type, it checks their histories for the presented pattern. Upon the positive results from the other histories, the system performs the appropriate action.

The following terminologies are referred very often in this work.

- Profile: all information that relates to a particular entity, e.g., information about the type of a person/user, location, time, duration, participants, etc. The profile can be updated and can evolve over time.

Category: the category of an entity is a set of other similar entities that share some basic characteristics, e.g., similar actions and histories of the same type or similar people with same interests.

Acquisition & Control layer: The sensor layer consists of two type sensors i.e. physical/hardware sensors and virtual/ software sensors. The physical/hardware sensors are Accelerometer, Gyroscope, Motion sensor, and RFID reader for person identification while the virtual sensors are Time reader, Temperature, Humidity, and Wind Speed from a web source while the location information is provided by the user by dividing the smart office/home dividing into multiple sub locations as shown in Figure 4.1. Both type of sensors feeds data to the upper layers for further processing. The frequency of the sensing is optimized to reduce the data processing load while keeping the performance of the system in an acceptable range. The sensing data is aggregated by the data fusion module at the same frequency from all the sensing sources and forwarded to the upper layer i.e. processing layer.

4.1. Data set and experimental configuration

In this section, experimental results on both indigenous virtual datasets prepared for university building and well known smart home dataset for activity recognition [85], [86] are shown and discussed.

4.2. Evaluation mechanism

The virtually created dataset contains the user id, name, location, timestamp, temperature, humidity, and wind speed as context data (independent variables) for the activity as an output data (dependent variable). For the university building data, the users are divided into four categories i.e. faculty, administration, students and guests. The actions of the system were considered as controlling the office electronic appliances including door and window. The actions are divided into three categories which mean that the user activities will lie in three different classes. The data is divided into training set and testing set by a ratio of 70 : 30.

For evaluation of the proposed technique, all computations were performed on AMD Athlon (tm) 64-bit X2 Dual Core 6000+ 3.10 GHz processor with 4 GB Physical Memory. However, the program was built using 32-bit compiler, which means full computational power is not utilized.

In order to test the system more extensively, data is divided into four sets of training data sets for every testing data set which are used to construct the rules for the context prediction model. The first training data set contains the information of the context histories of one type of users. The test data is generated from the same type of users. In the second training set the context information of the two types of user are used for testing the system. For the third test we extend the context information to three types of users. For the whole testing set the information of all the users were utilized for training. These data sets were formed in order to test the

dependability of the types of users on each other. Every time, new context data is added, the rules and prediction model are updated. This information is used to forecast the context of the users which are the part of the test dataset.

The accuracy of rules needs to be evaluated before applying on prediction. As we have a total of 9 parameters for our problem therefore our rule look like:

$$\{a_1 \wedge a_2 \wedge a_3 \wedge a_4 \wedge a_5 \wedge a_6 \wedge a_7 \wedge a_8 \wedge a_9\} \rightarrow \{C_i\}$$

The rules are evaluated using Laplace expected error estimate. The rule's accuracy estimation is given as:

$$LaplaceAccuracy(i) = \frac{x_i+1}{x_t+k}$$

Where x_t is the total number of instances covered by the rule, x_i is the positive instances covered and k is the total number of classes.

4.3. Result and discussions

In this section, The results obtained from the PCCP on indigenous test data sets are presented. The data collected from the accelerometer over a period of 60 minutes at a frequency of 50 Hz over a number of time. Multiple instances are recorded for the data which gives better training set for the system to understand the behavior f the activity. The mean of the collected data is shown in Figure 4.3 which depicts the accelerometer data by x-axis, y-axis and z-axis respectively.

Figure 4.4 shows the annotated data by through the kNN algorithm. The annotation is carried out on the raw data with respect to x-axis, y-axis and z-axis respectively. The activities are divided into 5 major categories i.e. Laying, Sitting, Climbing stairs, Standing, Walking. The accelerometer raw data is shown with respect to the activity annotation in different colors.

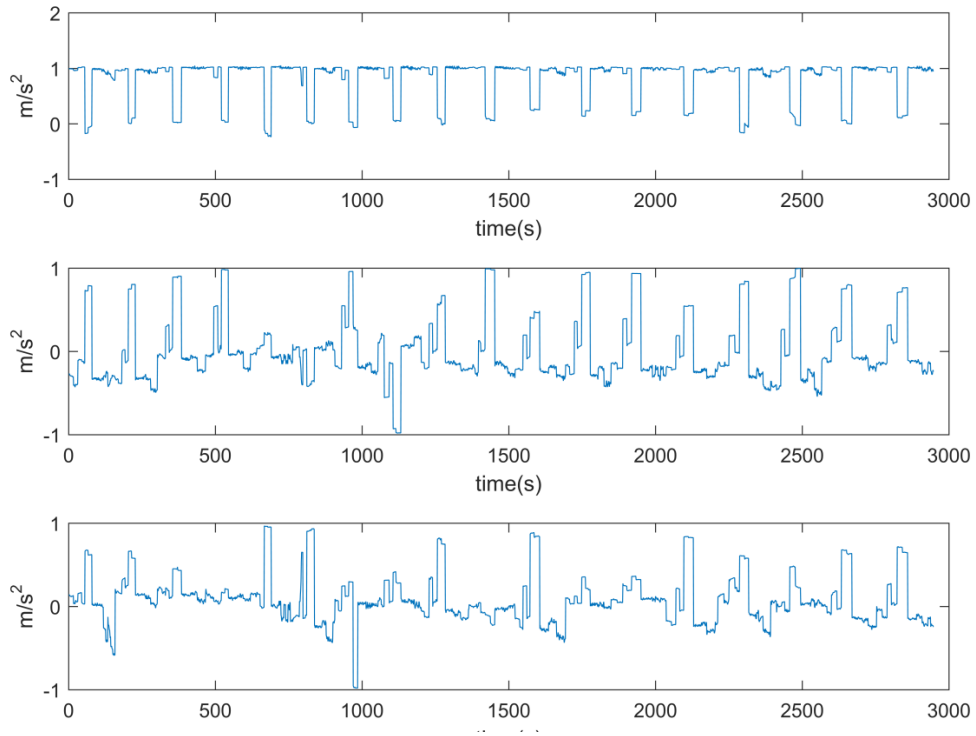


Figure 4.3: Mean of raw accelerometer data

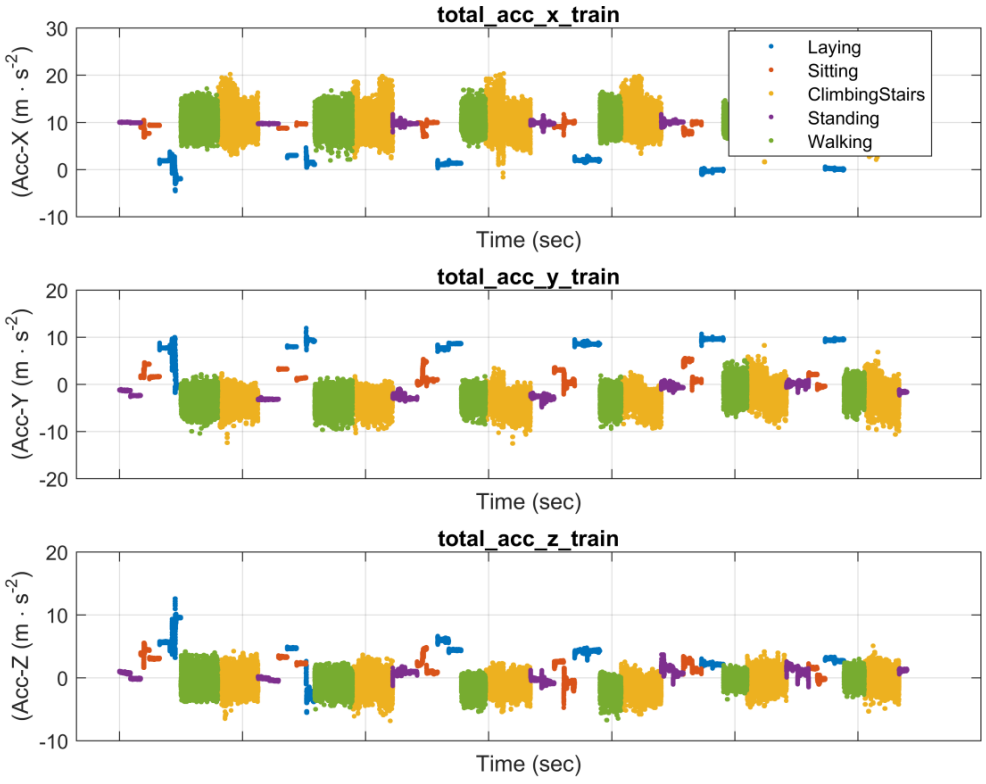


Figure 4.4: Raw accelerometer data with activity labels

Figure 4.5 shows the confusion matrix for kNN based activity annotation from the accelerometer data and gyroscope data. This data has been collected using "Matlab Classification Learner" toolbox which shows that results are promising for this type of annotation.

True Class						
	Laying	537 100%				
	Sitting		464 94.5%			
	ClimbingStairs			796 89.3%		
	Standing				511 96.1%	
	Walking					455 91.7%
		Laying	Sitting	ClimbingStairs	Standing	Walking
		Predicted Class				

Figure 4.5: Confusion matrix for kNN based activity annotation

Figure 4.6 shows the prediction results of the proposed system without collaboration with other users. The graph shows that increase in the number of instances results in the increases in accuracy of prediction, however, due to the limited amount of data, the accuracy cannot be increased after a certain level. The maximum accuracy of 87:89% achieved in this case.

Figure 4.7 shows the results of collaboration in our technique PCCP as compared to the CCP results presented in [8]. In [8] the authors used another technique called HOSVD for collaborative context prediction. However our technique of collaboration is different than that technique in many ways. We used the collaboration idea with a different implementation

technique. The CCP approach uses the data of all the users in the environment which take more time to converge. In our case we only use the information of the current user's category, due to this reason the predicted context accuracy is higher and the false positive ratio is very low in our case. This comparison of convergence is clearly depicted in Figure 4.9.

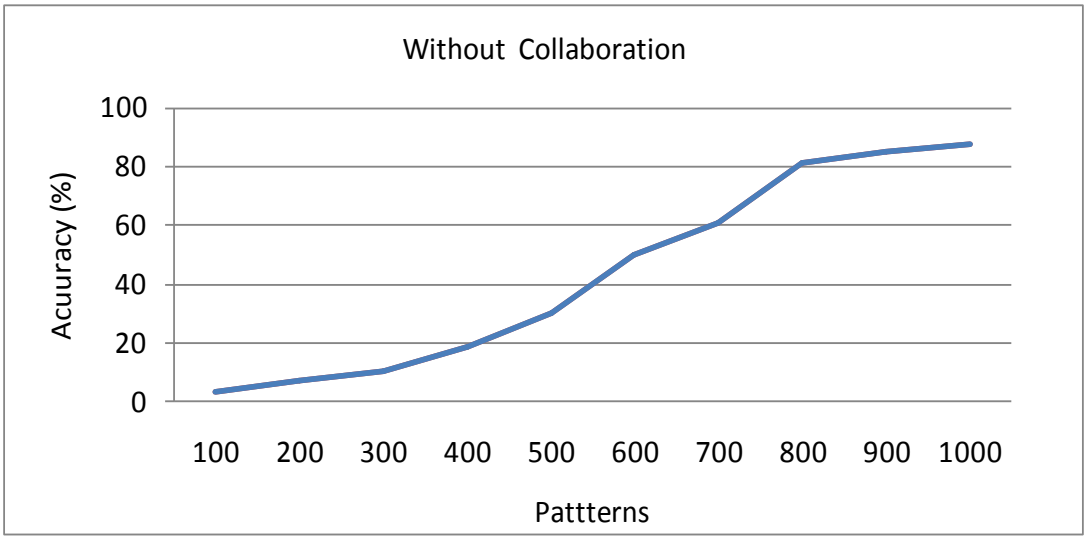


Figure 4.6: The accuracy graph of PCCP for the single user's environment

In [8] the authors used another technique called HOSVD for collaborative context prediction. However our technique of collaboration is different than that technique in many ways. We used the collaboration idea with a different implementation technique. The CCP approach uses the data of all the users in the environment which take more time to converge. In our case we only use the information of the current user's category, due to this reason the predicted context accuracy is higher and the false positive ratio is very low in our case. This comparison of convergence is clearly depicted in Figure 4.9.

In [8] the authors used another technique called HOSVD for collaborative context prediction. However our technique of collaboration is different than that technique in many ways. We used the collaboration idea with a different implementation technique. The CCP approach uses the data of all the users in the environment which take more time to converge. In

our case we only use the information of the current user's category, due to this reason the predicted context accuracy is higher and the false positive ratio is very low in our case. This comparison of convergence is clearly depicted in Figure 4.9.

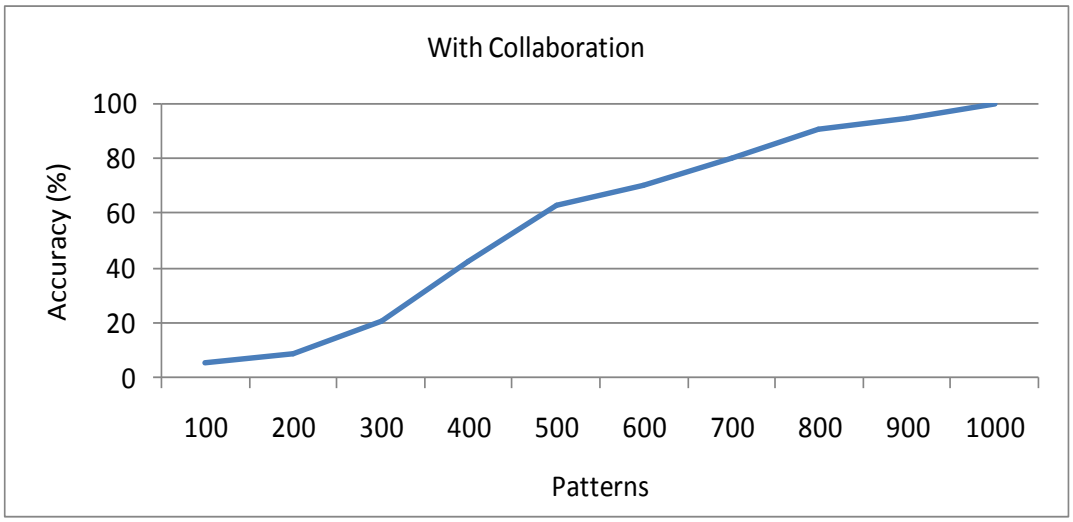


Figure 4.7: Accuracy of prediction by PCCP in collaborative environment

The users are divided into different categories and collaborate within the category. Besides, we use different technique to predict the context. The maximum accuracy achieved in the collaborative environment for PCCP reach to 100% when the number of instances reaches the maximum. The accuracy of the PCCP in collaborative environment is higher than without collaborating environment. This is because of the availability of information in the histories of other users.

In [8] the authors used another technique called HOSVD for collaborative context prediction. However our technique of collaboration is different than that technique in many ways. We used the collaboration idea with a different implementation technique. The CCP approach uses the data of all the users in the environment which take more time to converge. In our case we only use the information of the current user's category, due to this reason the

predicted context accuracy is higher and the false positive ratio is very low in our case. This comparison of convergence is clearly depicted in Figure 4.9.

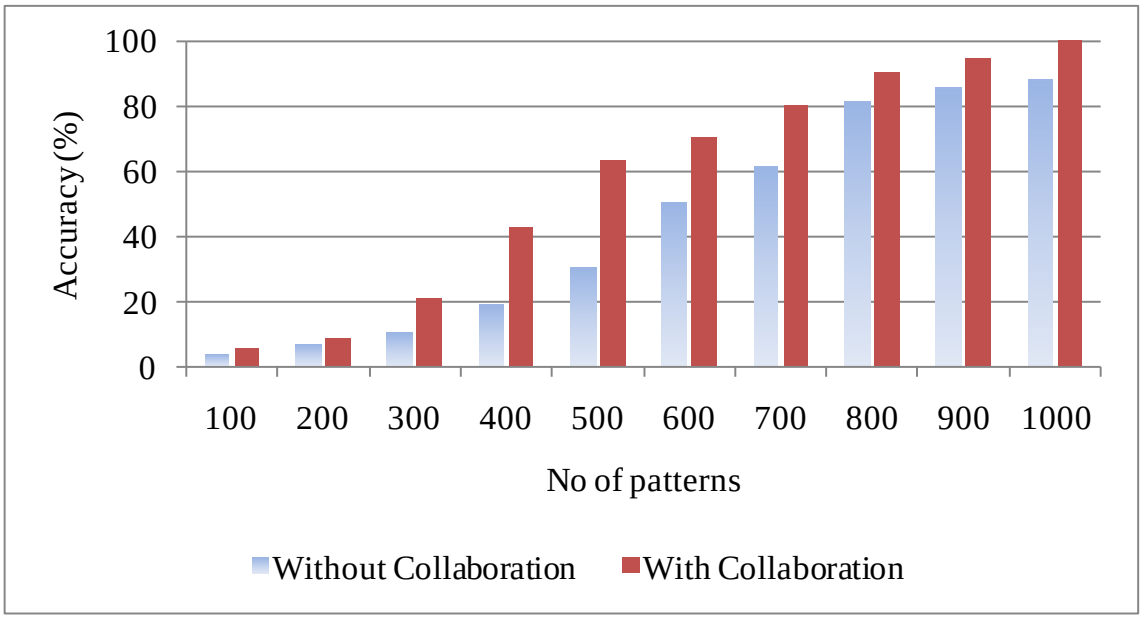


Figure 4.8: Comparison of prediction accuracy between with and without collaboration PCCP [8].

In Table 4.1, the results of the PCCP technique both with collaboration and without collaboration are presented.

Table 4.1: Comparison of with and without collaboration of the context prediction

No of Patterns	Accuracy (%) Without Collaboration	Accuracy (%) With Collaboration
100	3.56	5.67
200	6.98	8.76
300	10.2	20.77
400	18.56	42.76
500	30.12	63.21
600	50.23	70.42
700	60.98	79.87
800	81.43	90.43
900	85.32	94.76
1000	87.89	100

The results show that the future context can be predicted even when the context information is missing. The collaboration among other users can enhance the prediction accuracy. The PCCP is compared with the CCP and the results showed that our technique can produce better results in less amount of time than CCP as illustrated in Figure 4.9.

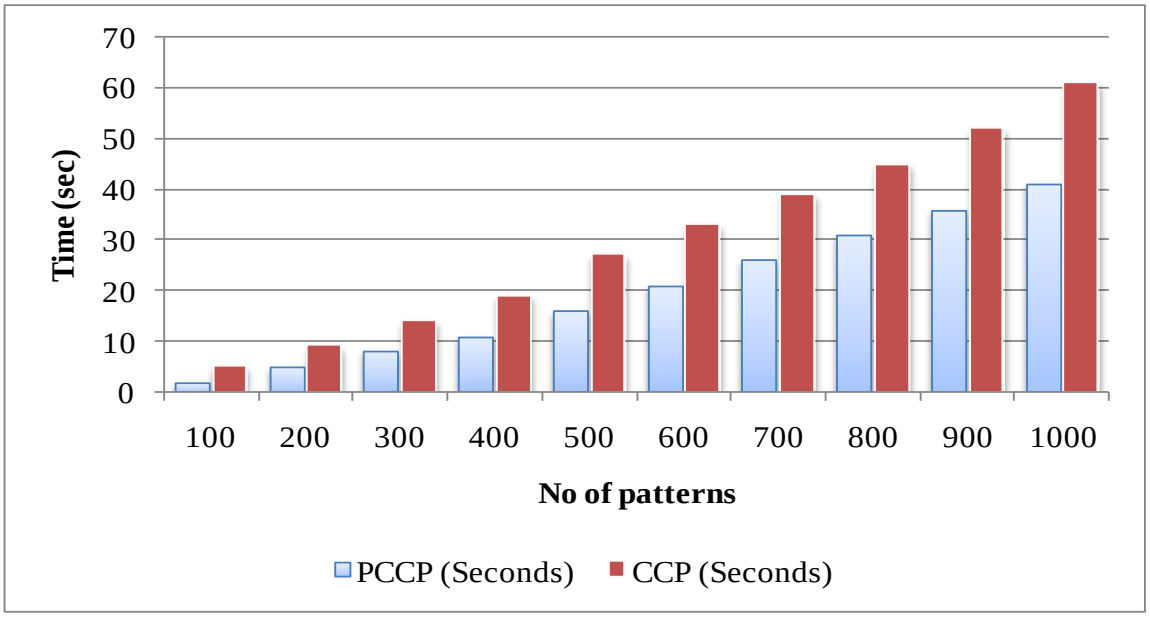


Figure 4.9: Time based comparison of CCP and PCCP

The results showed that the data from other users of the same profile can be used in case a user does not have sufficient information in his/her history. Furthermore, it is explored from the experiments that how the prediction accuracy for missing patterns affects the prediction time. The results are also compared with other collaborative context prediction techniques which do not use the profile information of the user.

Conclusion: A PCCP approach is proposed for context prediction in a multi user smart environment. The approach is focused in finding solution to the problem of missing or unknown context pattern in a multi-user environment. The context information of the other users is utilized to predict the context of missing or unknown pattern. Real world data is used for

evaluation of the proposed technique which shows that the PCCP can predict the context for unknown pattern in a reasonable time with a good accuracy rate. The PCCP is compared with the CCP and the results showed that our technique can produce better results in less amount of time than CCP. Due to the importance of the problem, more extensive experiments needs to be carried out with the state of the art prediction techniques like fuzzy logic to enhance the prediction level of the PCCP. The fuzzy can improve the results in a case where the perfect match does not find in the histories of all the users.

5. Experiment 2: Energy prediction and analysis in residential smart buildings

The advancement in sensing technologies open the opportunities to analyze and effectively solve the important problems of life. Residential and commercial buildings constitute the largest sector of South Korean electricity consumption i.e. 40% and is increased by 43% from 1990 to 2013 [87] and similar ratio in other countries. Due to this rapid increase in the electricity consumption in the commercial and residential buildings, it seeks proper attention to model and analyze the underline consumption patterns. This shall help in forecasting electricity demand and can lead to optimal electricity consumption.

The proposed CIoT architecture well suited to the energy management in a residential buildings due to the fact that contextual information of the user's behavior and environmental parameters heavily influence the electricity consumption in residential buildings. Besides, The collaborative nature of the architecture enhances the sharing the information among the different buildings and hence leads to avoidance of redundant data collection and processing. As buildings are the dominant source of energy consumption which seeks proper attention regarding energy management. The commercial and industrial buildings have been extensively studied for modeling and analyzing energy consumption. Traditional as well as data driven techniques have been electively applied for modeling and analyzing commercial and industrial buildings, however the residential buildings could only get the attention for traditional approaches. The data driven approaches are proved very effective for commercial and industrial buildings, however there is a gap of studies in utilizing them for residential buildings.

Recent studies [69][70], however brought the attention to the short term electricity consumption forecasting of the residential buildings. The first study [69] in this regard explored several machine learning techniques for a single family residential buildings, however they

ignored the multi-family residential buildings which are the dominant source of electricity consumption in urban areas. The short term electricity forecasting for a multi-family residential building has been studied in [70]. The authors utilizes a well-known data driven technique i.e. support vector regression and concluded that the data driven approaches can be used for forecasting of electricity consumption in multifamily residential buildings. While the limited work explored the multi-family residential buildings, there is still a need to explore more data driven approaches for short term forecasting of electricity consumption in multi-family residential buildings due to its importance and wide range benefits in energy optimization.

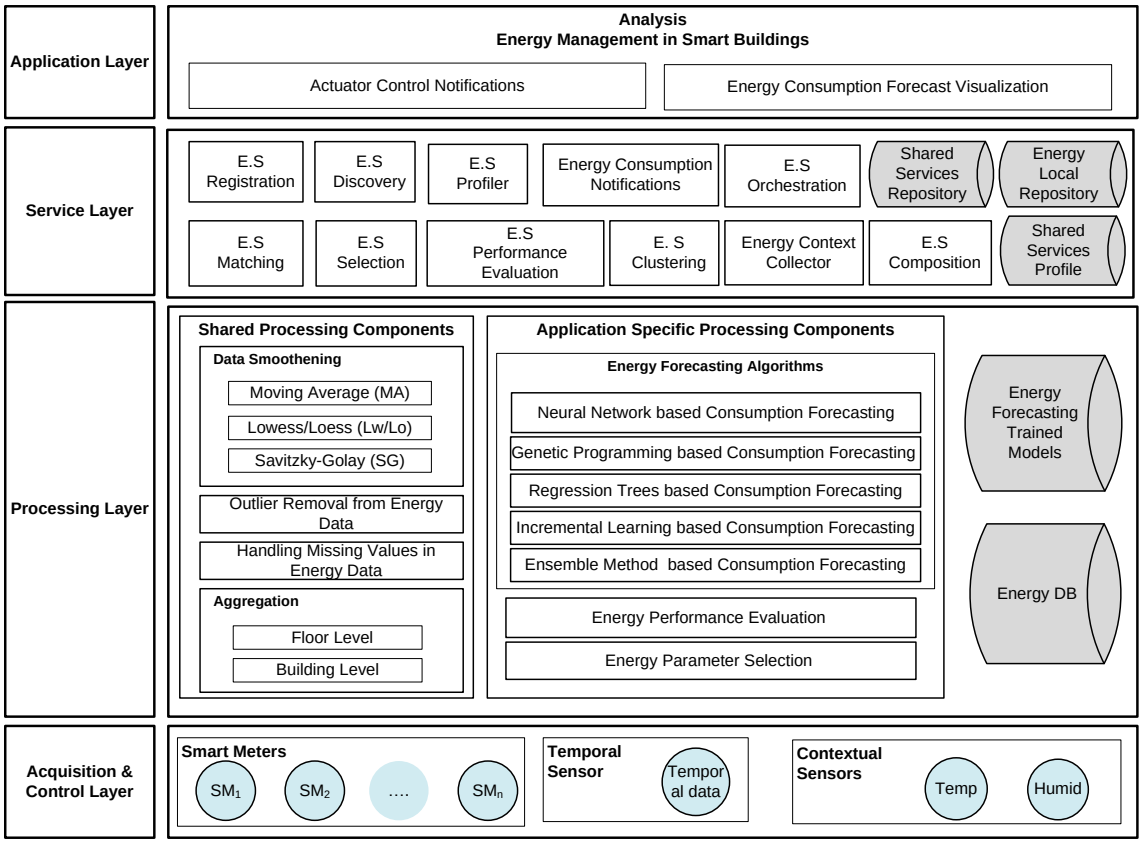


Figure 5.1: Component architecture for electricity consumption forecasting.

In this work, state of the art machine learning techniques i.e. Feed-Forward Neural Network (FFNN)[88], Genetic Programming (GP)[67], Classification And Regression Trees (CART)[89]

has been investigated and enhanced for forecasting the energy consumption of residential buildings in different cities of Republic of Korea. The impact of various pre-processing techniques is also studied. The forecasting is carried out at various spatial granularity level i.e. apartment, floor, and building. Also, the impact of spatial granularity on the prediction results is studied. The results shows the promising results and the approach is well suited for the prediction of energy consumption in the residential buildings. The layered system architecture of this work is shown in Figure 5.1. The layer wise description is given below:

Application Layer: This layer is the consumer layer of the processing carried out at the service and processing layers. The application layer carry out all the interaction with the users, handle the environment level details and proactively adapt the environment according to the user's context and the system's recommendation. As presented in Figure 5.1, two type of output is provided by this application i.e. actuator control notification and energy consumption visualizations. The actuator control notifications are in the form of text messages while the visualization is in the form of next 24 hour forecast and the reliability index in the form of Mean Absolute Percentage Error with respect to time.

Service Layer: Service layer provides the core functionalities of services registration, orchestration, composition and collaboration. The functionality of the components shown in Figure 5.1 is described in chapter 3, however, the specific detail of the components associated with this specific experiment is given below:

The data collected from the smart meters installed at the apartment level. The spatial unit is apartment, floor and building. The data aggregated at the floor level and the building level. This data is called as raw data. The raw data is passed to the preprocessing stage and there it is smoothened. After smoothing the day ahead forecast is carried out with multiple models. For day forecast the supervised and semi-supervised machine learning techniques. The environmental

data is shared as contextual information and also the building consumption behavior is collaborated with the clustered buildings. The functionality of these components are discussed in chapter 3, therefore, the key component i.e. forecasting techniques are discussed here in detail in this section.

The flow model of the proposed electricity consumption forecasting architecture is shown in Figure 5.2. It illustrates the flow of data among various component of the system. The components of each layer is depicted with the associated data and its functionalities.

The system starts reading the temporal, contextual, and real consumption data from their respective sensors. Different pre-processing is applied to the data i.e. normalization to the contextual data, smoothening, outlier removal & handling missing values are applied to the consumption data and day of week is extracted from the date. The data is fused at the Data fusion component and make patterns and consumption as two separate entities. The Data fusion component then passes the data to the predictive algorithms. The algorithms flow is described in next sections. The trained models are then evaluated on the test data. A decisive component is added to ensure the performance of the trained model is above a user set threshold. If the performance of the trained model is below the user threshold value then model is retrained with the updated parameter and training parameters by parameter selection module and training module. However, if the performance of the trained model is above the user set threshold then the model is selected for making the ensemble model. The ensemble component selects the candidate predictors and form an ensemble approach as described in the next section. The ensemble model is then evaluated on the test data and performance is compared with the base predictors. The model selection component then selects the model based on the performance values.

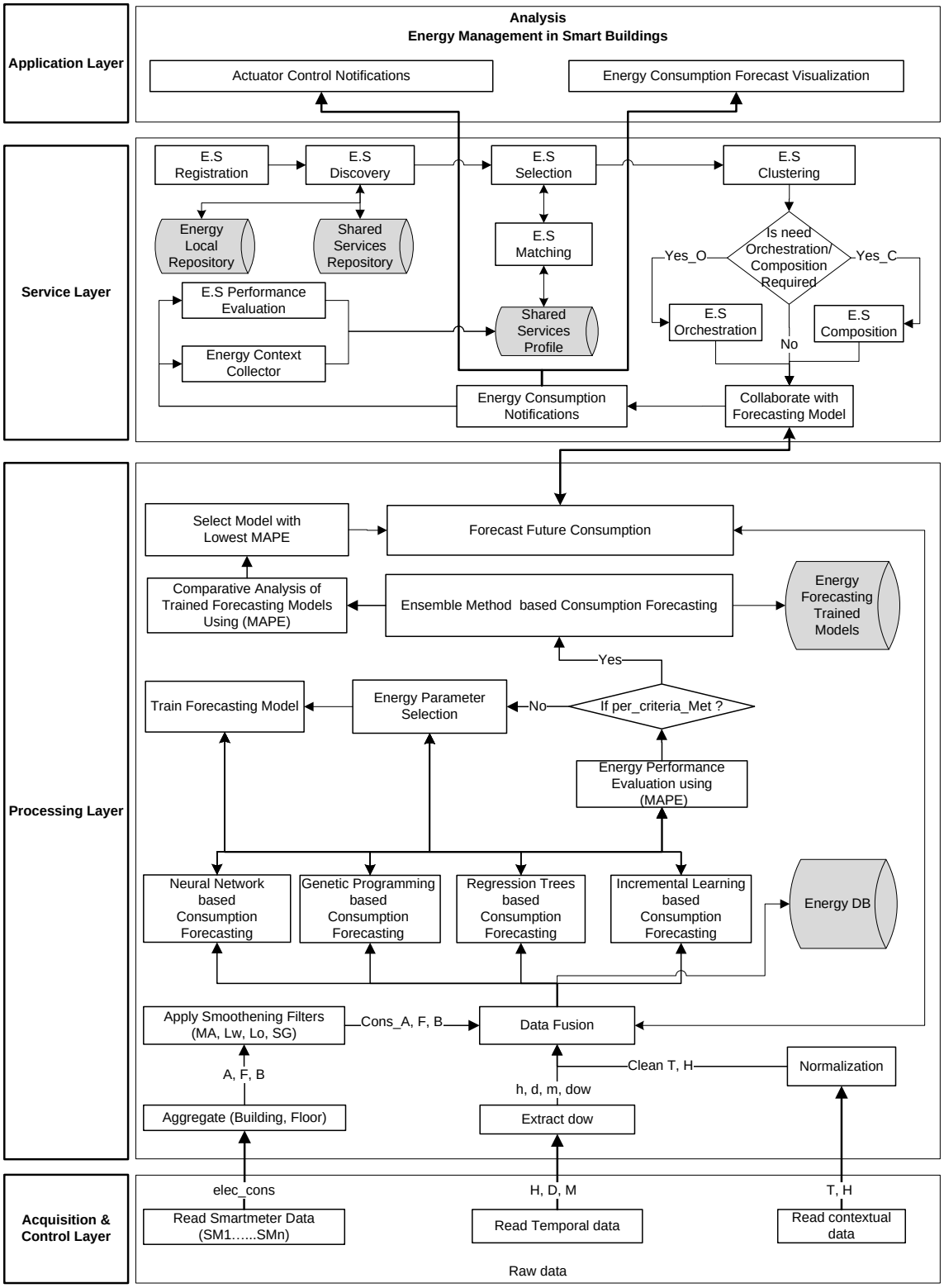


Figure 5.2: Flow model of the electricity consumption forecasting.

The best model is then enriched with the context prediction component which predicts the contextual data i.e. temperature & humidity and the future electricity consumption is forecasted. The actuator component then control the actuators based on the forecasted results.

Application Layer: The detail discussion of the component of this layer is given below:

Artificial Neural Network: ANNs are the most widely used artificial intelligence models in the application of building energy prediction. This type of model is good at solving non-linear problems and is an effective approach to this complex application. In the past twenty years, researchers have applied ANNs to analyze various types of building energy consumption in a variety of conditions, such as heating/cooling load, electricity consumption, sub-level components operation and optimization, estimation of usage parameters [60][61][62]. The ANN has many variants like MLPs, SOM, FFNN, BPNN and RNN, however for the purpose of this work FFNN has been used which has been proven best in load forecasting and commercial building consumption forecasting.

FFNN is elaborated for the day ahead consumption forecast in a residential building. FFNN can learn to approximate a function f that maps Rm → R without making assumptions about the relationship between the input and outputs. While a FFNN does not make assumptions about the inputs or outputs, it does require the user to define the model's structure, including the number of hidden layers and hidden units within the network and any other associated parameters. FFNN with a single hidden layer for function approximation has the following mathematical representation:

$$f(x) = \sum_{j=1}^{N} w_j \psi_j \left[\sum_{i=1}^{M} w_{ij} x_i + w_{io} \right] + w_{jo} \qquad \textbf{5-1}$$

Where N represents the total number of hidden units, M represents the total number of inputs, and Ψ represents the activation function for each hidden unit.

In this work, the following network structure is used:

- single hidden layer (with varying number of neurons)
- tan-sigmoid(x) as an activation function at the hidden layer
- linear function at the output layer

Levenberg–Marquardt algorithm [63] is used for training, which is one of the most popular training algorithm for FFNNs with gradient decent based method.

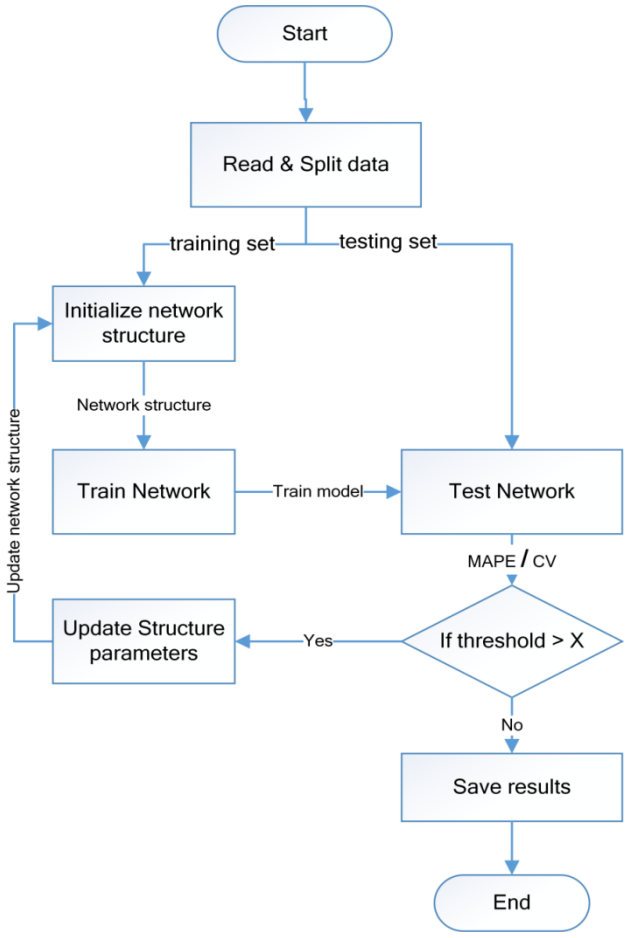


Figure 5.3 Flow diagram of the neural network based model learning

Figure 5.3 shows the flow diagram of the neural network based model learning for forecasting the electricity consumption. The model starts with the reading data as a blind model and continues to update the weights of the network to learn the invariant patterns from the data. The network structure is initialized with the model inputs along with the structural parameters.

The model inputs are considered in consistence with the work presented in the [69][70] which are defined as:

$$X(t) = [y(t-1); y(t-2); T(t); s; sh; ch]$$

where y(t - 1) and y(t - 2) represent known electrical consumption values for the previous two time steps, T(t) is the current temperature, S (t) is the current solar flux, s is an indicator variable that denotes weekend/holiday or weekday, sh is the sine of the current hour and ch is the cosine of the current hour. Due to non-availability of data presented in the equation 4, we define the model input for testing on our real data set as:

$$X(t) = [y(t-30; \ldots; y(t-1); h; d; dow; m; T(t); H(t)]$$

where y(t - 30); ... ; y(t - 1) is the electricity consumption for the last 30 days, h is set of hours, d is the set of days, dow is the day of week, m is month of year, T(t) is set of temperature values, and H(t) is the set of humidity values in the given time period. The 30 days data is used for training due to the reason that the daily pattern exhibits the same behavior which leads to a good forecast.

We observe during the experimentation that, if the training data is short then it leads to a low accuracy in forecasting the 24 hour ahead consumption. Once the model is trained to forecast a single step ahead consumption, our system presents the next input pattern and forecast the consumption. As long as the error between the forecasted value and the actual value remains in the acceptable range, the system continuously forecast the consumption. When the error over

pass the acceptable range, the system triggers the training module and retrains the model with the most recent observed patterns. This phenomenon is shown in Figure 5.4.

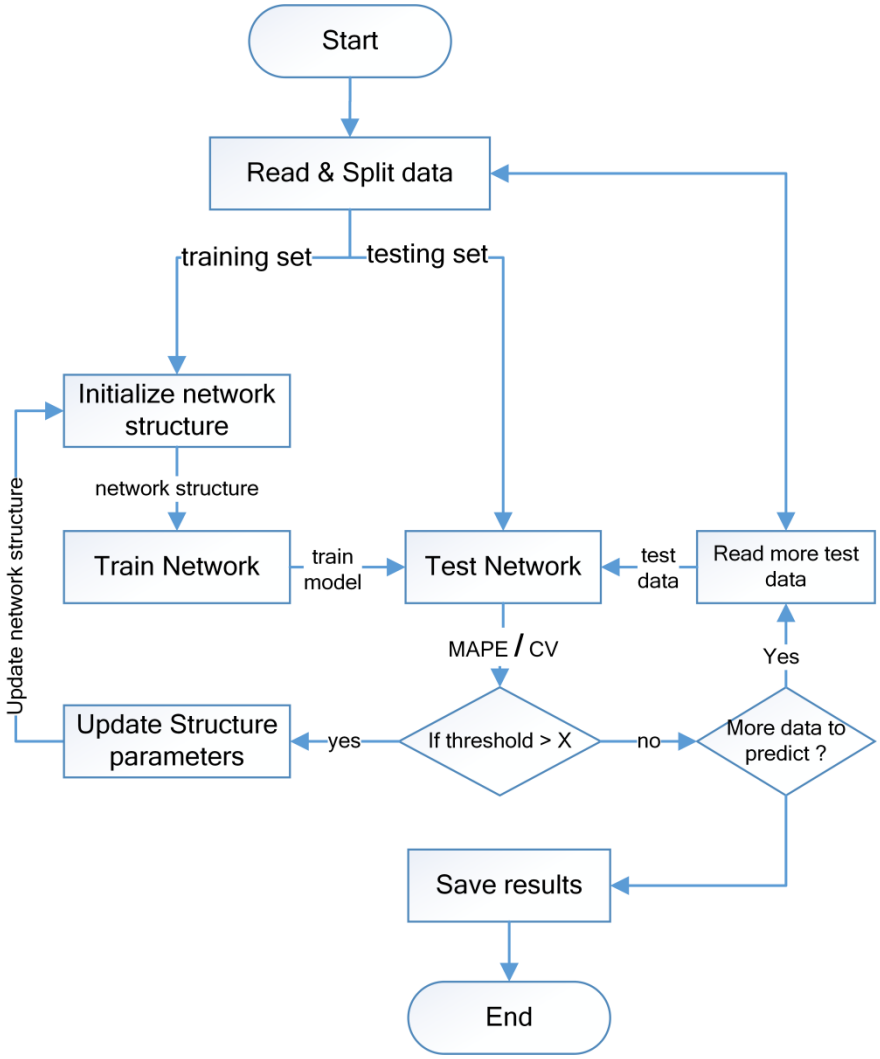


Figure 5.4: Flow diagram for the training of model for single multi-step (hours) ahead forecast

This work is the extension of the work presented in [69][70] which are single-step prediction approaches and predicts one hour ahead consumption, however our model is a single-step as well as multi-step approach as our single step is 24 hour ahead.

Classification And Regression Tree (CART): In recent times, there has been increasing interest in the use of CART analysis. CART analysis is a tree-building technique which is

different from traditional data analysis methods. In a number of studies, CART has been found to be quite effective for creating decision rules which perform as well or better than rules developed using more traditional methods [64][65]. In addition, CART is often able to uncover complex interactions between predictors which may be difficult or impossible using traditional multivariate techniques.

The methodology of CART used in this work is outlined here. The enhanced recursive partitioning algorithm [66] is utilized for CART. This enhanced recursive partitioning improves the performance of the CART and also improves the convergence time. The algorithm work in a step-by-step process by which a decision tree is constructed by either splitting or not splitting each node on the tree into two child nodes.

The model input are defined as:

$$X(t) = [y(t-30; \ldots; y(t-1); h; d; dow; m; T(t); H(t)]$$

where y(t - 30); ... ; y(t - 1) is the electricity consumption for the last 30 days, h is set of hours, d is the set of days, dow is the day of week, m is month of year, T(t) is set of temperature values, and H(t) is the set of humidity values in the given time period. The 30 days data is used for training due to the reason that the daily pattern exhibits the same behavior which leads to a good forecast.

The following procedure is adopted to determine how to split node *t*. For all predictors $X(t) = [y(t-30; \ldots; y(t-1); h; d; dow; m; T(t); H(t)]$.

a) Compute the weighted mean-square error (WMSE) of the response in node *t* using

$$\varepsilon_t = \sum_{j \in T} w_j (y_j - \bar{y}_t)^2$$

w_j is the weight of observation j, and T is the set of all observation indices in node t.

when the weights are not specified, then $w_j = 1/n$, where n is the sample size.

b) Sort x_i in ascending order. Each element of the sorted array is a splitting candidate for prediction.

c) Determine the best way to split node t using x_i by maximizing the reduction in MSE (ΔI) over all splitting candidates. That is, for all splitting candidates in x_i:

   i. split the observations in node t into left and right child nodes (t_L and t_R, respectively).

   ii. calculate ΔI. Consider that for a particular splitting candidates, t_L and t_R contain observation indices in the sets T_L and T_R, respectively.

d) Choose the candidate that yields the largest MSE reduction.

The algorithm will splits the predictor variable at the cut point that maximizes the MSE reduction.

The size of tree has been controlled using the following procedure:

- To handle the maximum number of splits, the current layer splits all the nodes at time and the number of branch nodes are counted. If the number of branch nodes exceeds α (the max level threshold), the following procedure is followed:

  1. Find that how many branch nodes at the current layer must be unsplit so that there are at most α branch nodes.

  2. Calculate the impurity gain for each node and sort accordingly.

  3. Unsplit the number of leafs with minimum gain.

  4. Return the grown decision tree at the layer.

Genetic Programming: Genetic Programming (GP) is an evolution means of representing artificial intelligence action by means of programs, and its basic idea is the Theory of Evolution obeying to the natural rules of selecting the superior and eliminating the inferior, the survival of the fittest[67]. Genetic programming inherits the basic idea of Genetic Algorithm (GA), but contrarily it is different from Genetic Algorithm.

Genetic Programming starts with randomly creating an initial search space of individual chromosomes (computer programs) consist of the functions and terminals. Each individual in the search space has its own fitness. Fitness of the individual is evaluated by minimizing the value of MAPE using the equation:

$$Fitness\_function = Minimize\left(\frac{1}{N}\sum_{i=0}^{N}\frac{y_i - \bar{y}_i}{y_i}\right)$$

As many as functions that reflect the essence of variables and load value of each time point are selected, for the relationship between them cannot be determined well and truly beforehand. Here the functions are F = {+, −, *, /, sin, cos, sec, tan, ln, exp, sqrt} and the terminals are T = {h, d, dow, m, ρ}, where h is the hour, d is the day, dow is the day of week, m is the month, and ρ is the electricity consumption.

Procedure of GP for electricity consumption forecasting

Inputs:

{ $X(t) = [y(t-30; \ldots; y(t-1); h; d; dow; m; T(t); H(t)]$, [+, −, *, /, sin, cos, sec, tan, ln, exp, sqrt] , NoofRun, PopSize, electricity consumption}

Outputs:

Trained Model

Procedure GP

Run: =0;

FOR I: =1 TO NoofRun DO

```
BEGIN
  Gen: =0
  Generate Initial Population Randomly
  WHILE NOT (Terminate Condition for Run)
   BEGIN
        Evaluate fitness of each individual (MAPE);
        FOR J: =1 TO PopSize DO
          BEGIN
               Op: =Select Genetic Operator;
                BEGIN
                  Select two individuals based on fitness;
                  Perform Crossover with Probability Pc;
                   J: =J+1;
                   Insert two offspring into new Population;
                END;
            END;
      Gen: =Gen+1;
    END;
  Designate Result for Run;
  Run: =Run+1;
END
END.
```

Incremental learning of NN (Self training): Self-training is a commonly used method for semi-supervised learning. In this method a classifier is first trained with the sample of labeled data. Then the classifier is used to classify the unlabeled datasets. Normally the most assured unlabeled data, along with their predicted labels, are appended to the training set. The classifier is re-trained with the new data and the process is repeated. This process of re-training the classifier by itself is called self-teaching or bootstrapping.

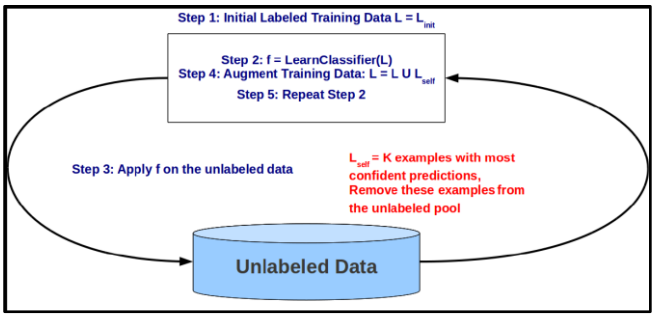


Figure 5.5: Conceptual model of self-training

The problem of multi-step electricity consumption forecasting (long-term forecasting) is an ideal problem to be solved with the incremental learning. The reason is that the data collected is for one year while we want to extend the forecast for a longer period, so the available consumption data for training is shorter than the forecasting data. In this work, the data is recorded for 1 year i.e. 2010 and a long term forecasting is carried out for the ease of energy management companies. Therefore, a novel incremental model is devised to forecast a long term forecasting using a small amount of training data. This is achieved with the below procedure:

The train model continues to forecast the step ahead consumption and evaluate the error difference between the predicted consumption and actual consumption. The threshold value is set by the user to tell the model the acceptable error of the model. Once the model's error crosses the acceptable range, the system retrains the model as shown in Figure 5.6.

The incremental learning approach for neural network is described as follow with a graphical abstraction in Figure 5.5 .

- Input: $X(t) = [y(t-30; \ldots; y(t-1); h; d; dow; m; T(t); H(t)]$
- Output: Long-term forecast
- Idea: Train, predict, re-train using best predictions, repeat
- Step 1: x=the initial set of consumption data (i.e. 720 hours)

- Step 1.a: y=the prediction time is 24 hours
- Step 2: x+y (the initial consumption data and the predicted data)
- Step 2.a: predict (next 24 hours consumption)

Continue till the required number of days.

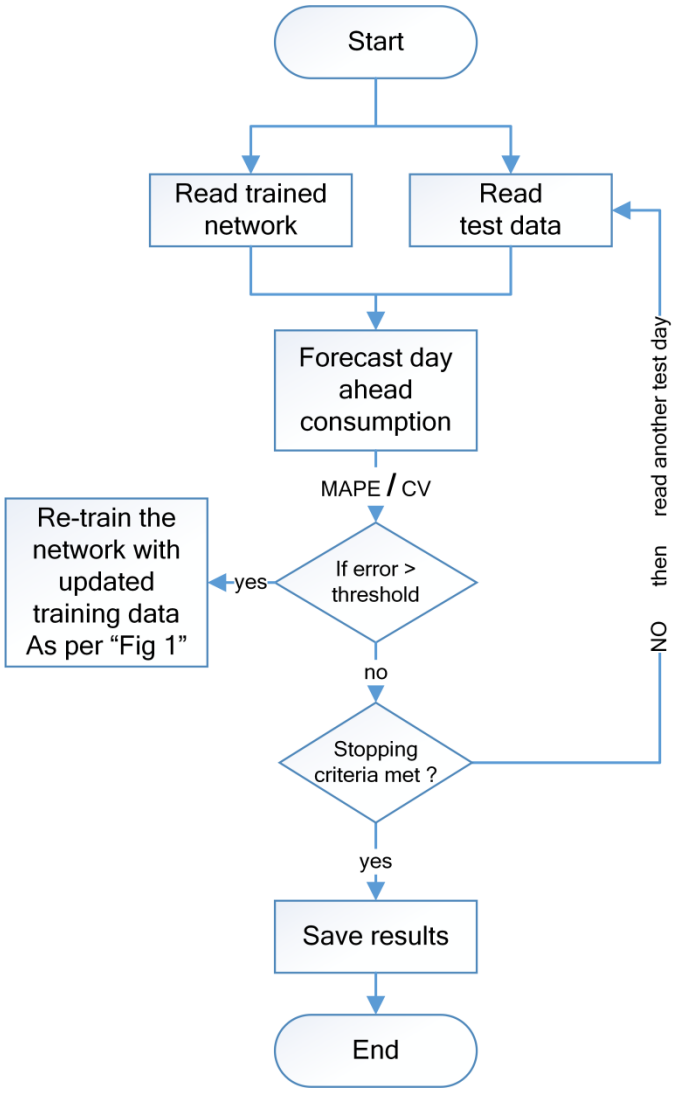


Figure 5.6: Flow diagram for incremental learning of neural network for multi-step forecasting

Ensemble Model (Co-training): An ensemble approach based on co-training semi-supervised learning technique is proposed for the enhanced performance of the electricity

consumption forecasting. Co-training assumes that each example is expressed using two different data sets that provide different but matching information about the instances. Preferably, the two views should be conditionally independent and each view is adequate. Co-training first learns a individual predictors/regressors for each view using training data. The most reliable predictions of each predictors/regressors on the testing data are then used to iteratively construct additional training data.

A co-training style semi-supervised regression algorithm, i.e. COREG, is proposed in [68]. This procedure uses two regressors where one regressor labels the unlabeled data for the other regressor. The COREG style of co-training mechanism is adopted for carrying out experiments, however the based regressors are different. We used the supervised learning techniques (GP, RTREE, and FFNN) described in the previous section as the base prediction techniques.

The ensemble model is developed from the supervised techniques as discussed above. The ensemble is then extended for unlabeled data as in a co-training mechanism as shown in Figure 5.7.

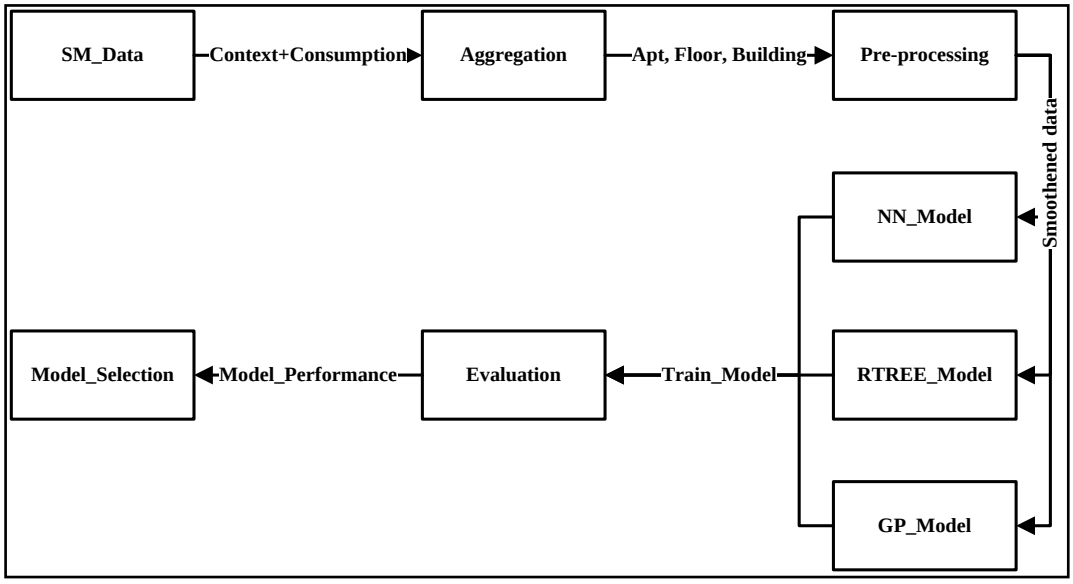


Figure 5.7: Ensemble Model

Ensemble Model procedure

Inputs:

{ $X(t) = [y(t-30; \ldots; y(t-1); h; d; dow; m; T(t); H(t)]$, electricity consumption}

Outputs:

Ensemble Model

Ensemble Procedure

Begin

Read SM_Data (smart meter data)

Read Weather data (Context)

Aggregate (consumption_data) //floor level, building level

Preprocess (consumption_data) // apartment level, floor level, building level

Train NN_Model

Train GP_Model

Train RTREE_Model

Evaluate (NN_Model, GP_Model, RTREE_Model) on test data

Choose Best Model

Continue

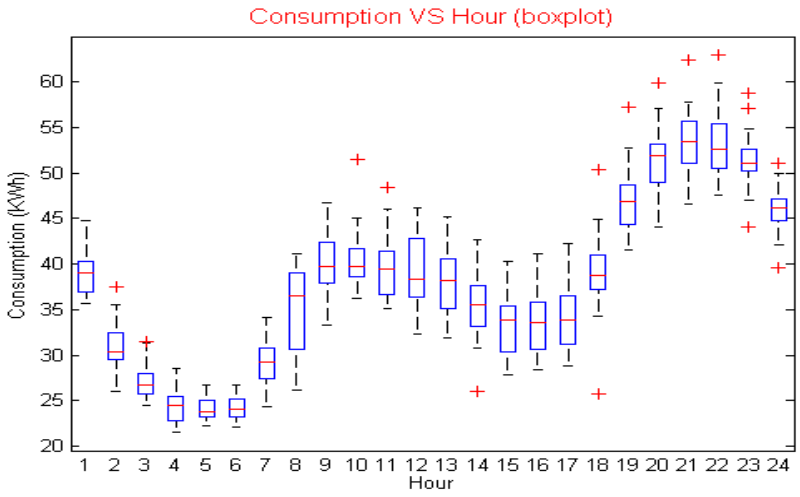
If termination criteria is satisfied

End

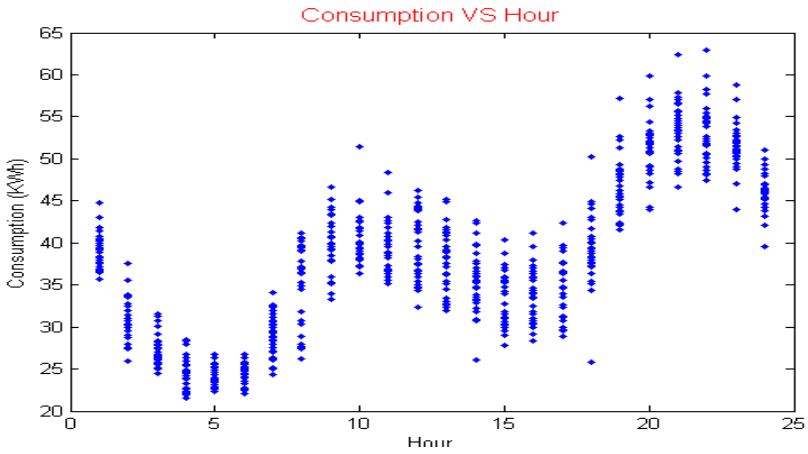
5.1. Data set and experimental configuration

The data used for experimentation of this work is collected from different building in urban area of South Korea. All the buildings are multi-family residential buildings where each apartment contains bedroom, kitchen, bathroom and a living area. The data is collected through smart meters installed at the individual residential apartments in a building at different locations of a city for a period of 1 year (Jan 2010~Dec 2010).

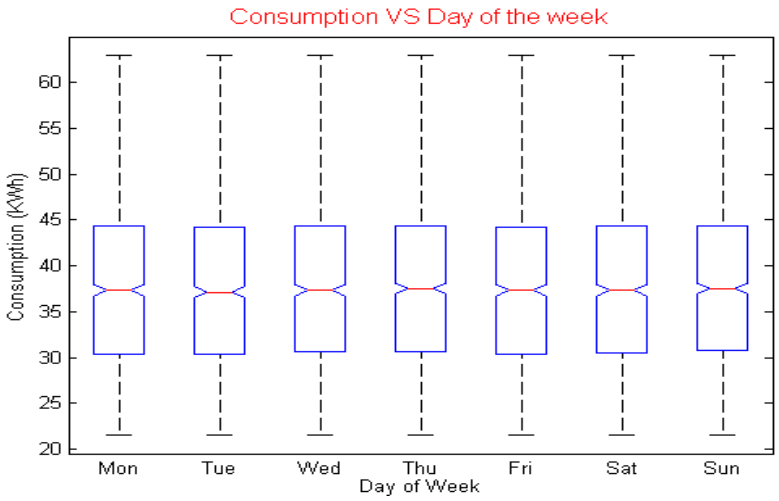
The size of training data is varied in order to evaluate the impact of training size and to cope with seasonal changes. Besides the environmental factors i.e. temperature and humidity is also considered for experimentation; however we didn't find any impact of weather on the electricity consumption. The data is divided into three different training sets i.e. T1 (1 month long), T2 (3 months long), and T3 (6 months long). The testing set is day ahead which is 24 hours long. Figure 5.8 shows the structure of data.



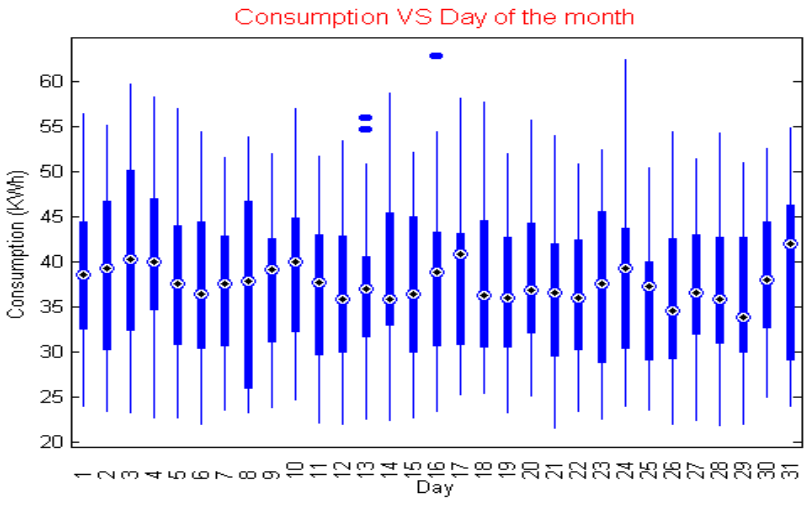
Sub Figure a: Box plot of consumption vs hours



Sub Figure b: Plot of consumption vs hours



Sub Figure c: Box plot of consumption vs day of week



Sub Figure d: Plot of consumption vs day of month

Figure 5.8 Basic data statistics

5.2. Evaluation mechanism

We evaluate the performance of our model with Mean Absolute Percentage Error (MAPE) which is consistent with [4][6].

$$MAPE = \frac{1}{N}\sum_{i=0}^{N}\frac{y_i - \bar{y}_i}{y_i}$$

where y_i is the actual consumption and $\bar{y}_i$ is the predicted value and N is the total number of consumption hours.

5.3. Results and discussion

The predictive results for a whole year is recorded based on increasing the size of training data. The training is started from a 01 month data which is used to predict the day ahead consumption. The training size is incremented by a day and continued till the end of the year. The purpose of this kind of training was twofold: investigate the impact of size of training data and validating the predictive results.

5.3.1. Data Pre-processing

Several preprocessing techniques are applied to remove the noise and smooth the consumption patterns. Data is collected through smart meters and aggregated on hourly basis. Some form of random variation is always present in a collection of data taken over time. An often used technique to reduce the effect of random variation is called smoothing. Smoothing always involves some form of local averaging of data such that the nonsystematic components of individual observations cancel each other out. When this method is applied properly it reveals more clearly the underlying trend, seasonal and cyclic components.

Figure 5.9 shows the effect of moving average on a one day aggregated consumption of a one building data from Seoul city. The window size for the figure is 3 which is a small window size and does not affect the original data too much. The smaller the window size the lesser the smoothing effect, and lesser changes in the original data.

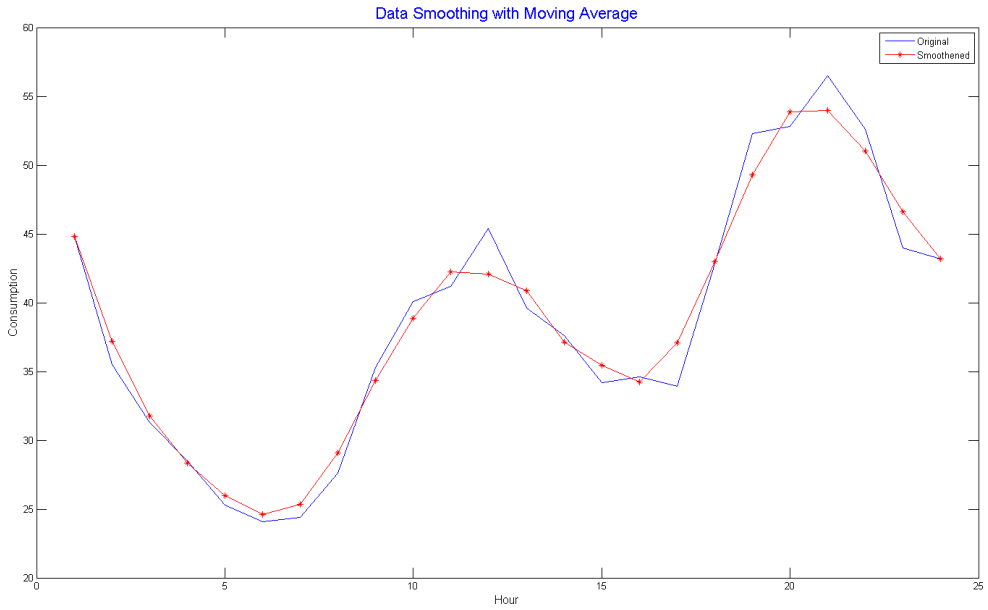


Figure 5.9 Effect of moving average on one day consumption (window size =3)

Figure 5.10 shows the effect of loess smoothing on the one day electricity consumption. The span size is 0.4. The greater the span size the smoother the data results. Therefore the optimal value of span size is found with trial process.

Figure 5.11 shows the effect of lowess smoothing on the one day consumption. For lowess the span size is 0.25 which gives the better results with less amount of information lost or less amount of smoothing. The greater the smoothing, the greater the information loss and hence the greater the noise added into the signal. If the signal has some noise the smoothing will remove it but if there is no noise then the smoothing will add noise to the signal. Therefore the effect should be optimal for both cases.

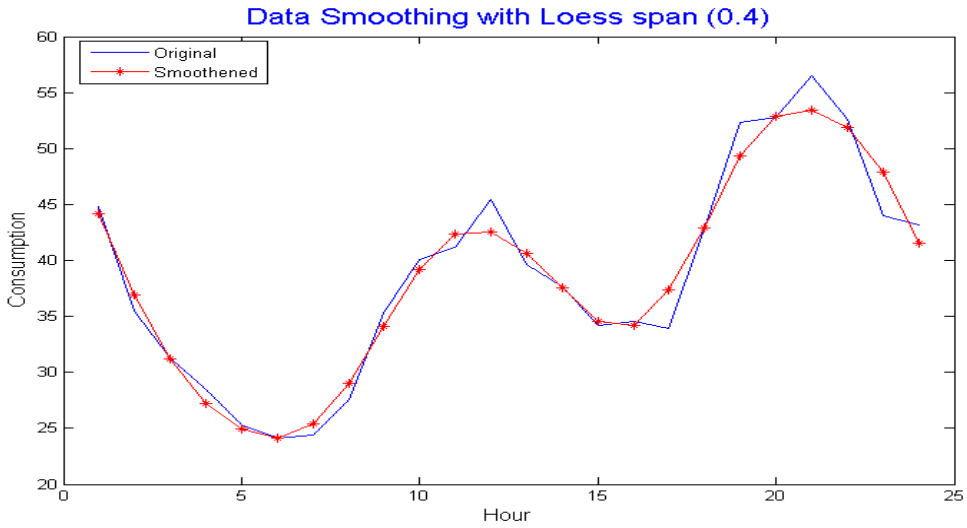


Figure 5.10: Loess smoothing effect on a one day consumption

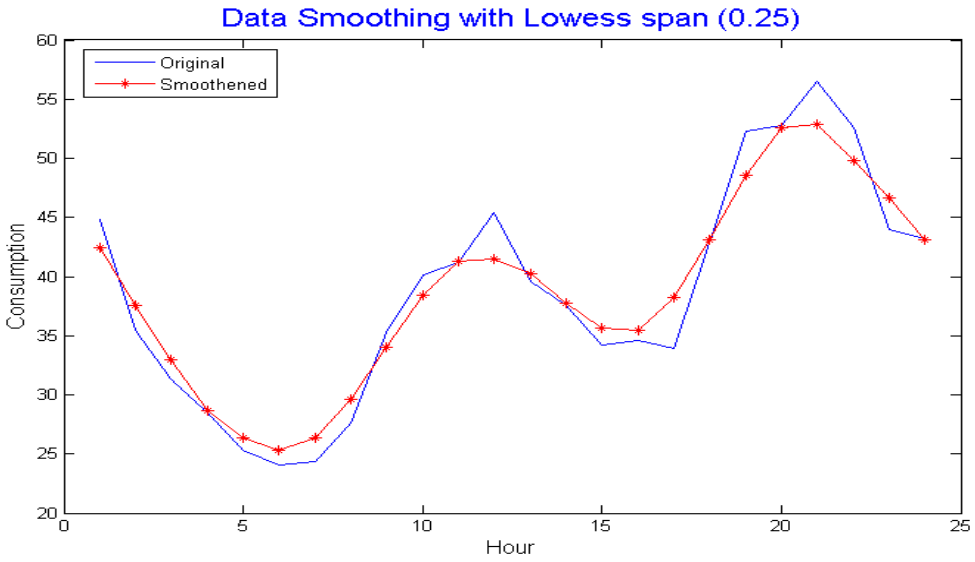


Figure 5.11: Lowess smoothing effect on a one day consumption

Figure 5.12 shows the effect of robust loess smoothing on the one day consumption. The span size is 7 which is absolute value for the span. It is evident from the figure that the consumption curve is smoothened by removing the small peak and preserving the shape of the data.

Figure 5.13 shows the effect of robust lowess smoothing on the one day consumption. The

optimal span size after trivial process is 5. The effect of robust lowess is smaller than robust loess as it is shown in both figures.

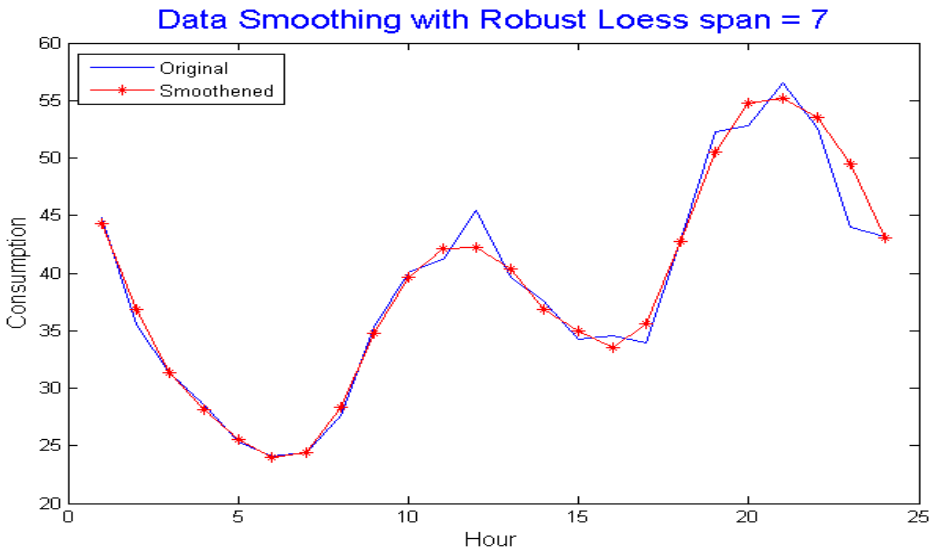


Figure 5.12: Effect of robust loess on one day consumption

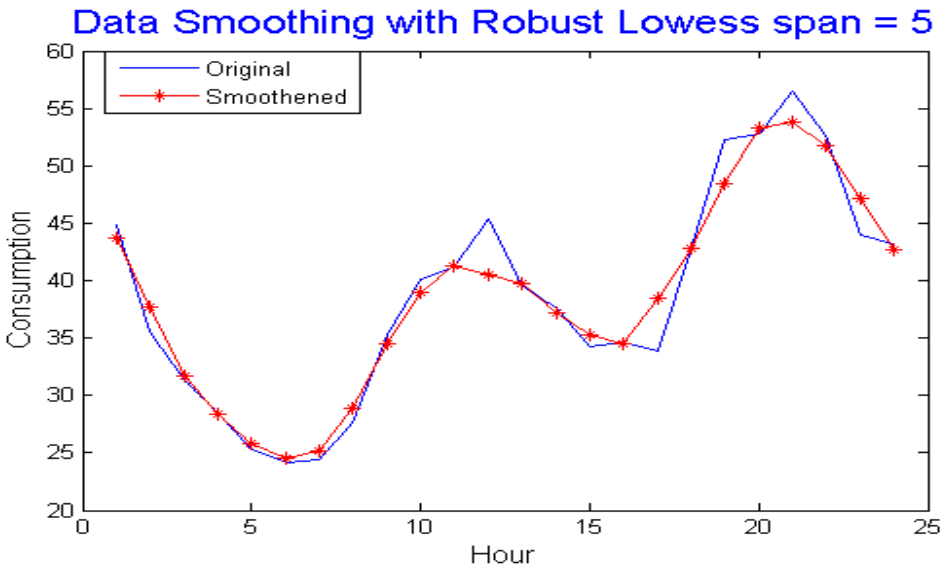


Figure 5.13 Effect of robust lowess on one day consumption

Figure 5.14 shows the smoothing effect of savitzy golay filter on one day consumption. It can be observed from the figure that the curve is more smoothened than the other filters.

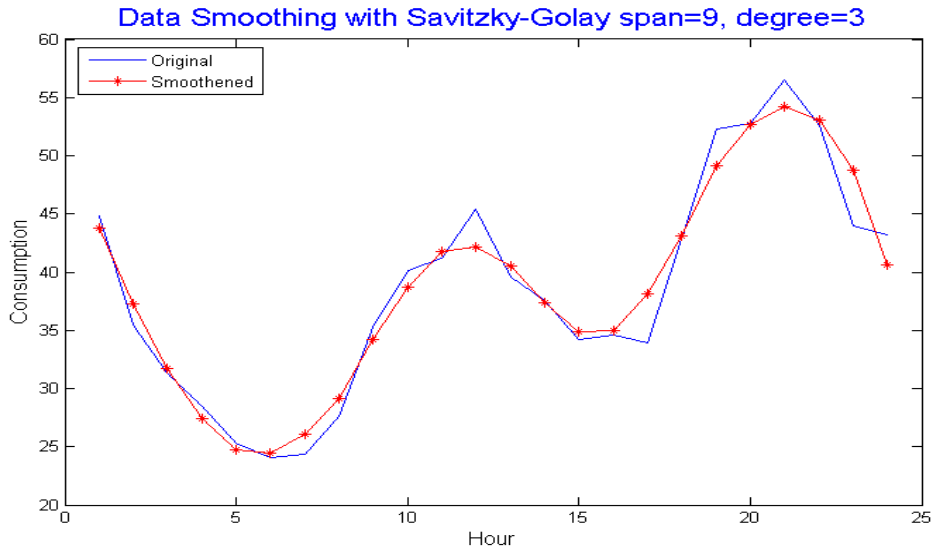


Figure 5.14: Effect of Savitzky golay smoothing on one day consumption

Figure 5.15 shows the comparative plot of the above discussed smoothing filters. The figure shows that all the filters smoothened two prominent peaks i.e. in the mid of day and one in the night. The rest is all smoothened the curve and made is easy for comparison purposes.

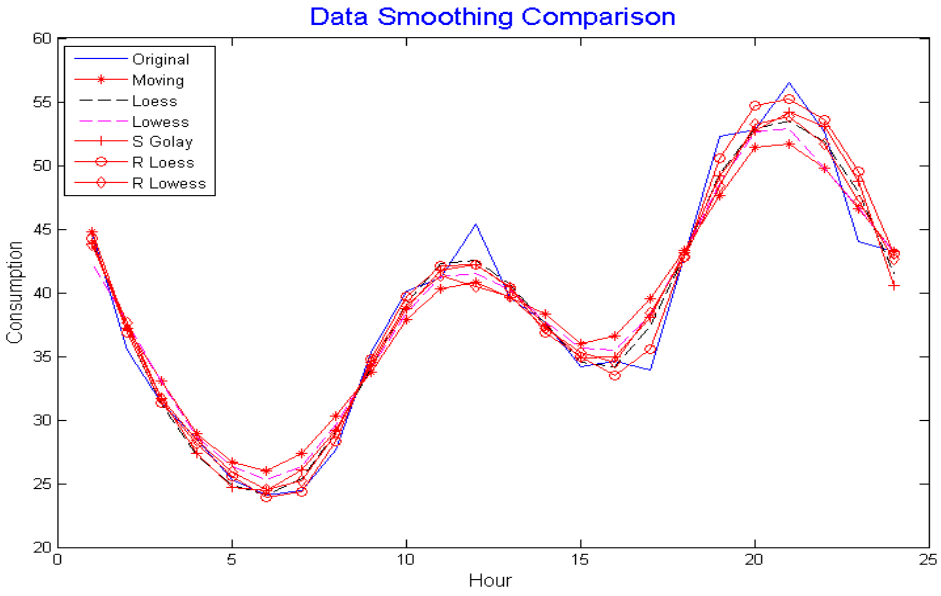


Figure 5.15: Effect of Smoothing techniques on a single day consumption

Figure 5.16 shows the error diagram of the above discussed smoothing filters with original

data. All the filters are applied to one day consumption. The filters are applied to training data for machine learning and predicted results are compared with original consumption.

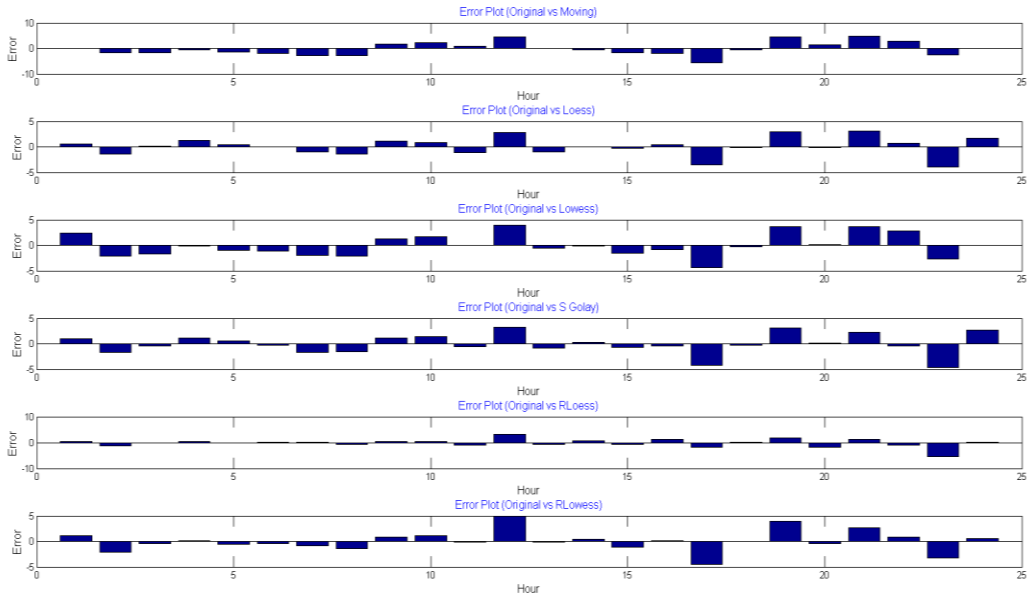


Figure 5.16: Error plot for smoothing filters with original data

5.3.2. Neural Network in processing layer

Figure 5.17 shows the plot of predictive results for a best predicted day in a yearlong prediction for a whole building scenario. It is evident from the figure that our model does not over fit to the data which results in a better predictive accuracy. The aggregation of whole building apartment's consumption decreases the impact of user behavior which has been considered as one of the major factor influencing the energy consumption in a residential building [90]. This decrease in variability results in a better accuracy in comparison to the floor and apartment level prediction.

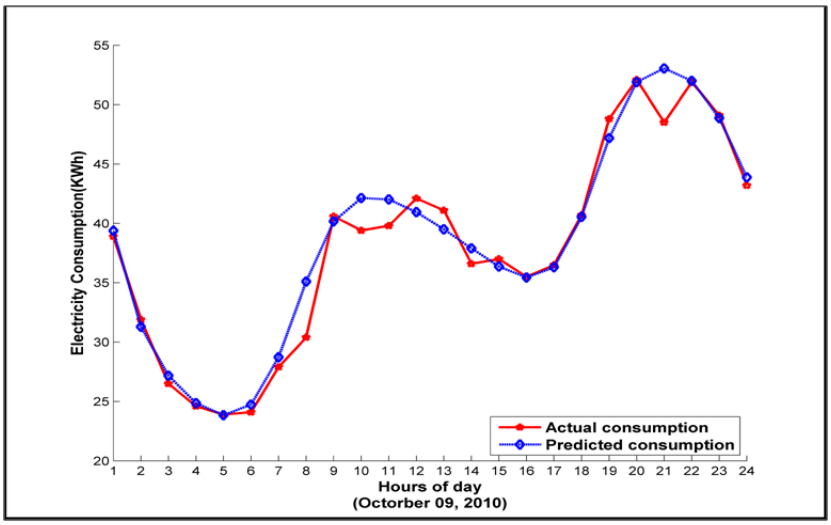


Figure 5.17: Building scale hourly forecasting result of a sample day with CV=2.91

The Figure 5.18 and Figure 5.19 shows the plot for floor level and apartment level predictive results for sample days. There were 4 apartments in a floor which causes the impact of variability on the prediction results at the floor level. The variability in the apartment level is the highest among the three levels and hence leads to high CV value. Table 5.1 shows the comparison of the CV values for the building, floor, and apartment level prediction results. The best, worst, and mean CV values are shows for different days. The building level CV value is less by 50% than the previously reported results in [69][70].

Table 5.1: Coefficient of Variation (CV) values at different spatial granularity level

Granularity Level	***CVbestday***	***CVworstday***	***CVmean***
Apartment	16.73	53.47	32.68
Floor	14.13	40.74	23.88
Building	2.91	11.66	5.23

The predictive results reported in [70] shows that the best prediction is achieved at the floor level which is contrary to the results we achieved for our neural network model. Our model

produced the best predictive results at building level which is consisted with the concept that the greater the variability in the consumption patterns, the lower the prediction accuracy. It is also evident from the Table 5.1 that the difference in prediction error for the apartment level and floor level is small which is due to the reason that each floor consist of only 4 apartments.

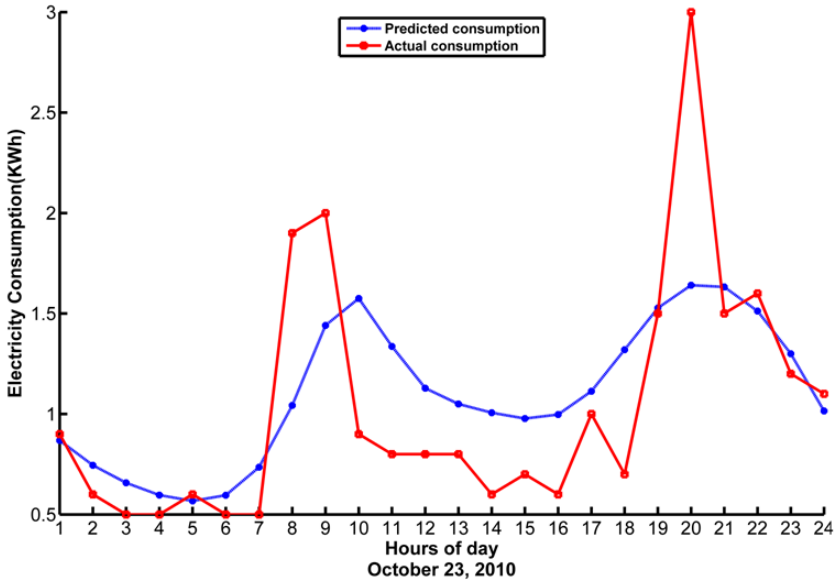


Figure 5.18: Floor scale hourly forecasting result of a sample day with CV =14.13

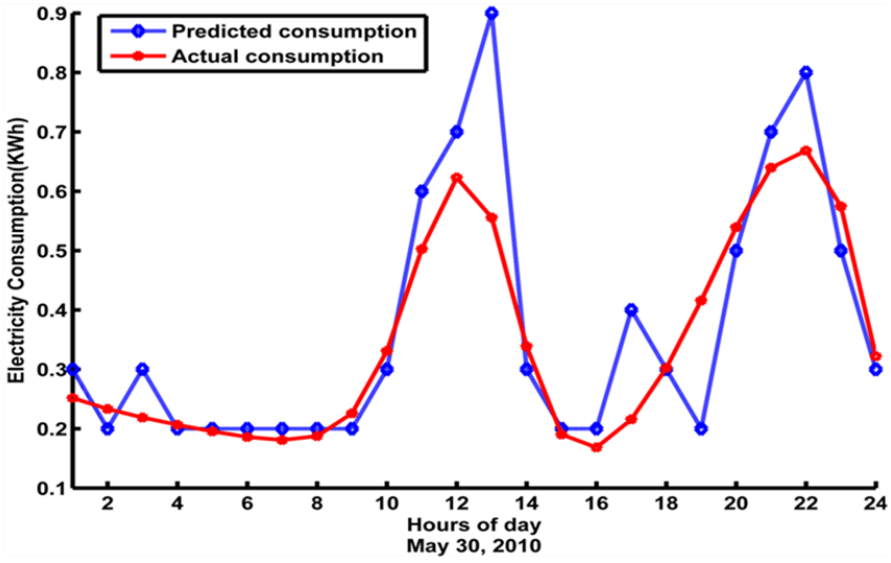


Figure 5.19: Apartment scale hourly forecasting result of a sample day with CV =16.73

Table 5.2: Results of NN based forecasting

Data	NN_Best Case	NN_Average Case	NN_Worst Case
RLoess15	5.47	6.34	6.81
MA3	5.64	6.72	7.48
Loess15	5.82	6.87	7.54
Rlowess7	5.83	6.55	7.37
Lowess5	6.16	6.92	7.26
Rlowess5	6.26	6.64	7.02
SGolay93	6.37	6.91	7.42
Original	6.45	6.83	6.99
RLoess10	6.47	6.80	7.08
Lowess7	6.67	7.14	7.43
Loess7	6.78	7.14	7.40
SGolay113	6.86	7.30	7.95
Loess5	6.91	7.24	7.76
SGolay73	7.04	7.42	8.10
MA5	7.12	7.31	7.46
Rlowess10	7.51	8.18	8.72
Lowess11	7.77	8.30	9.04
RLoess20	8.37	8.72	9.36
MA7	8.38	9.11	9.63

Table 5.2 shows the mean absolute percentage error (MAPE) for the day ahead forecasting using neural network. Since the neural network produces different results every time it retrains, therefore, we evaluated the performance over a 10 runs of neural network over the best selected parameters. The results from the best run, average run and worst run is shown in Table 5.2. The table shows the impact of pre-processing on the results of the neural network. As it is evident

from the table that the RLoess15 produces the best set of results, however due to the additional complexity of the RLoess15, we recommend the use of MA3 which is a simple moving average of size 3. The MA3 is simple and does not cost the computation power as is required for the RLoess15. However, in a performance critical situation the consumer can choose the preprocessor according to his ease and priorities.

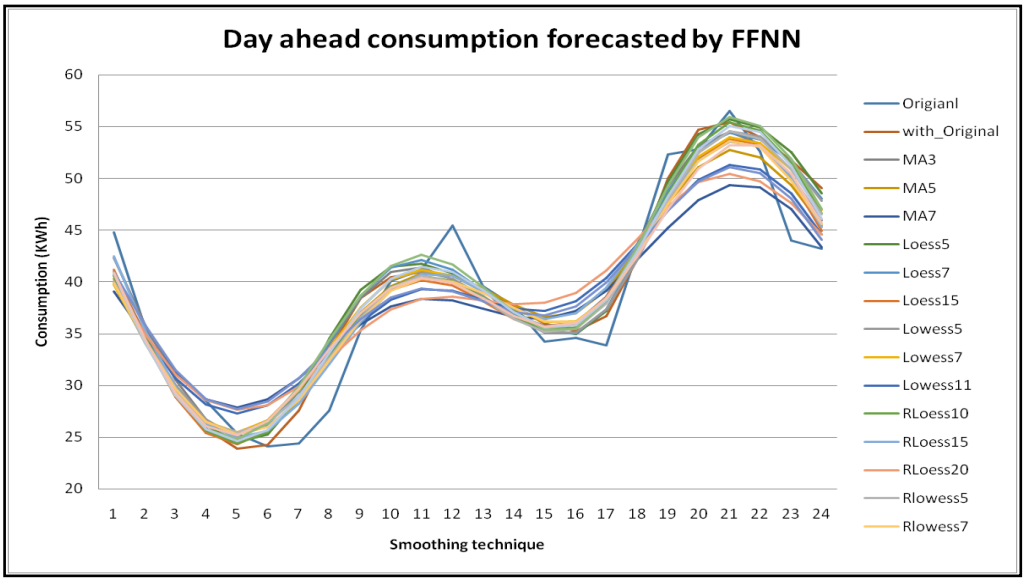


Figure 5.20: Day ahead electricity consumption prediction result by FFNN

The comparative graph of the result of the best day ahead forecast of NN with different pre-processing techniques is shown in Figure 5.20. The pre-processing does impact on the performance of the forecasting. The abrupt changes in the data is removed through the pre-processing which leads to a better accuracy from the neural network.

Figure 5.21 shows the comparison of best case, average case and worst case results of the neural network over different pre-processing techniques.

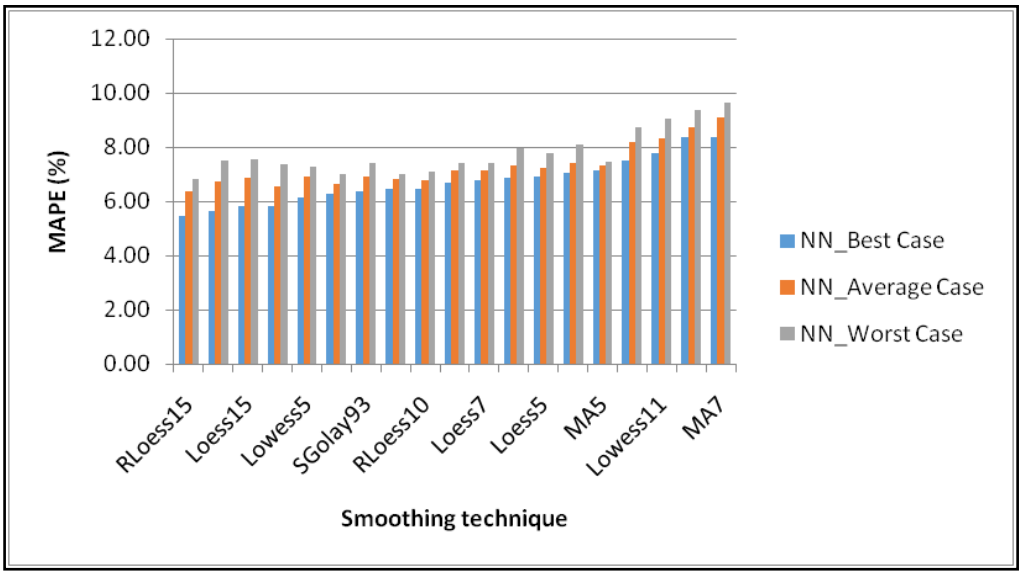


Figure 5.21: MAPE for day ahead consumption prediction using FFNN

5.3.3. Classification And Regression Tree (CART) in processing layer

Figure 5.22 show the tree build through the induction process for a one month training data. The data used for this experiment is aggregated data of one building from Seoul city. The tree show the branches bases on the 4 input pattern parameters i.e. hour, day, day of week, and month. The input parameters are called x1, x2, x3 and x4 respectively. The leap nodes are the electricity consumption which is the final value for an input pattern.

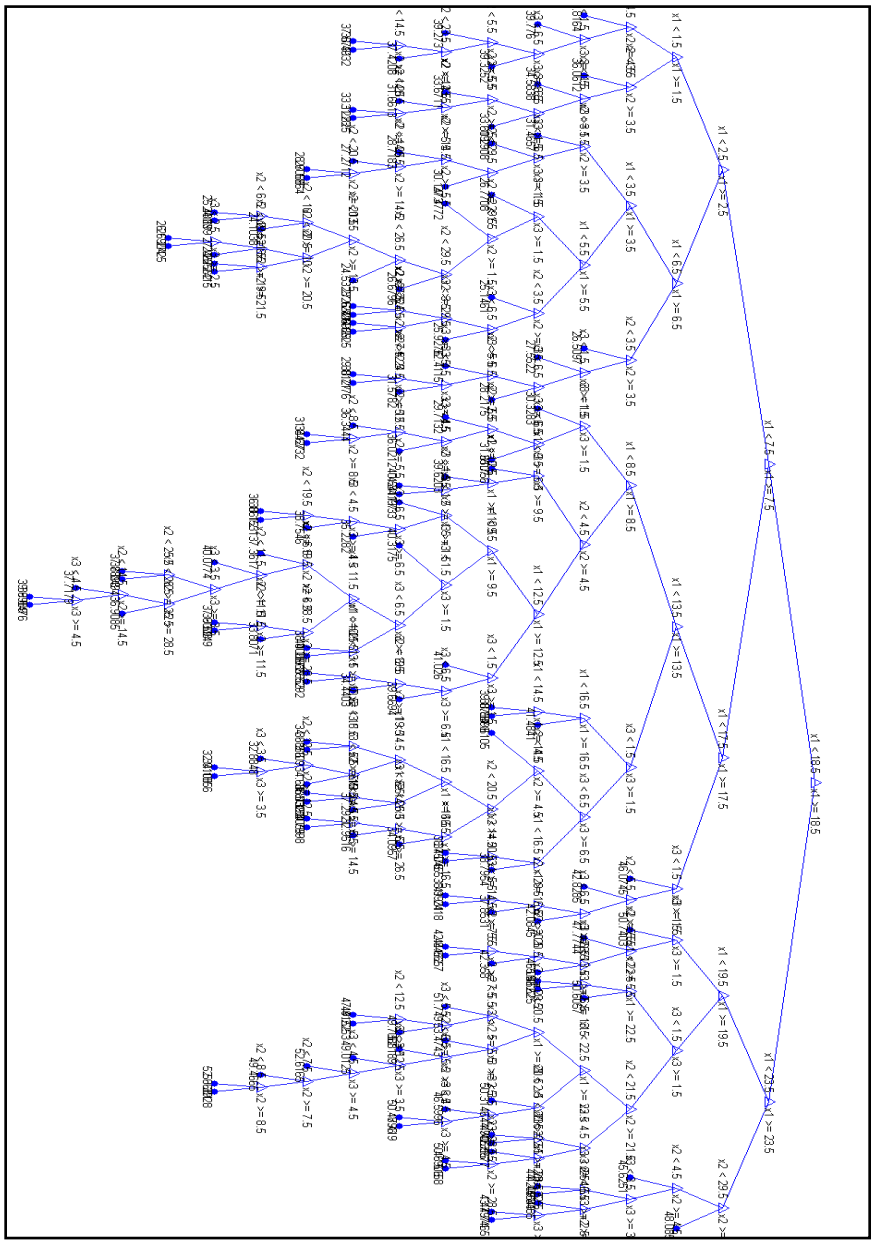


Figure 5.22: Cart tree on a one month training data

Table 5.3 shows the sample set of rules. The CART produces these rules from the tree during the induction process and uses these rules for forecasting the day ahead consumption. The process of prediction is called deduction from the regression tree.

Table 5.3: Sample rule set of cart on one month training data

1 if x1<18.5 then node 2 elseif x1>=18.5 then node 3 else 37.7195	2 if x1<7.5 then node 4 elseif x1>=7.5 then node 5 else 34.1557	3 if x1<23.5 then node 6 elseif x1>=23.5 then node 7 else 48.4108
4 if x1<2.5 then node 8 elseif x1>=2.5 then node 9 else 29.6704	5 if x1<17.5 then node 10 elseif x1>=17.5 then node 11 else 37.0099	6 if x1<19.5 then node 12 elseif x1>=19.5 then node 13 else 49.2142
7 if x2<29.5 then node 14 elseif x2>=29.5 then node 15 else 44.3939	8 if x1<1.5 then node 16 elseif x1>=1.5 then node 17 else 35.8488	9 if x1<6.5 then node 18 elseif x1>=6.5 then node 19 else 27.1991
10 if x1<13.5 then node 20 elseif x1>=13.5 then node 21 else 36.6364	11 if x3<1.5 then node 22 elseif x3>=1.5 then node 23 else 40.7448	12 if x3<1.5 then node 24 elseif x3>=1.5 then node 25 else 45.415
13 if x3<1.5 then node 26 elseif x3>=1.5 then node 27 else 50.164	14 if x2<4.5 then node 28 elseif x2>=4.5 then node 29 else 44.2666	**15 fit = 48.0856**
16 if x2<4.5 then node 30 elseif x2>=4.5 then node 31 else 38.6762	17 if x2<3.5 then node 32 elseif x2>=3.5 then node 33 else 33.0215	18 if x1<3.5 then node 34 elseif x1>=3.5 then node 35 else 26.5241
19 if x2<3.5 then node 36 elseif x2>=3.5 then node 37 else 29.8988	20 if x1<8.5 then node 38 elseif x1>=8.5 then node 39 else 37.4376	21 if x3<1.5 then node 40 elseif x3>=1.5 then node 41 else 35.4347

22 fit = 46.0745	23 if x2<5.5 then node 42 elseif x2>=5.5 then node 43 else 39.9248	**24 fit = 50.7403**

Table 5.4: MAPE of RTREE

Data	**MAPE**
RTREE_Original	4.678
RTREE_MA3	**4.088**
RTREE_MA5	5.570
RTREE_MA7	8.221
RTREE_Loess5	4.670
RTREE_Loess7	4.176
RTREE_Loess15	5.728
RTREE_Lowess5	4.292
RTREE_Lowess7	4.894
RTREE_Lowess15	8.588
RTREE_RLoess10	4.641
RTREE_RLoess15	6.292
RTREE_RLoess20	10.943
RTREE_RLowess5	4.303
RTREE_RLowess7	5.193
RTREE_RLowess10	8.601
RTREE_SGolay73	4.106
RTREE_SGolay93	4.15
RTREE_SGolay113	5.274

Figure 5.23 shows the results from the CART. The results are recorded for all the smoothing filters and sorted from lowest MAPE to highest MAPE. The lowest MAPE is for the highest accuracy while the highest MAPE is for the worst accuracy.

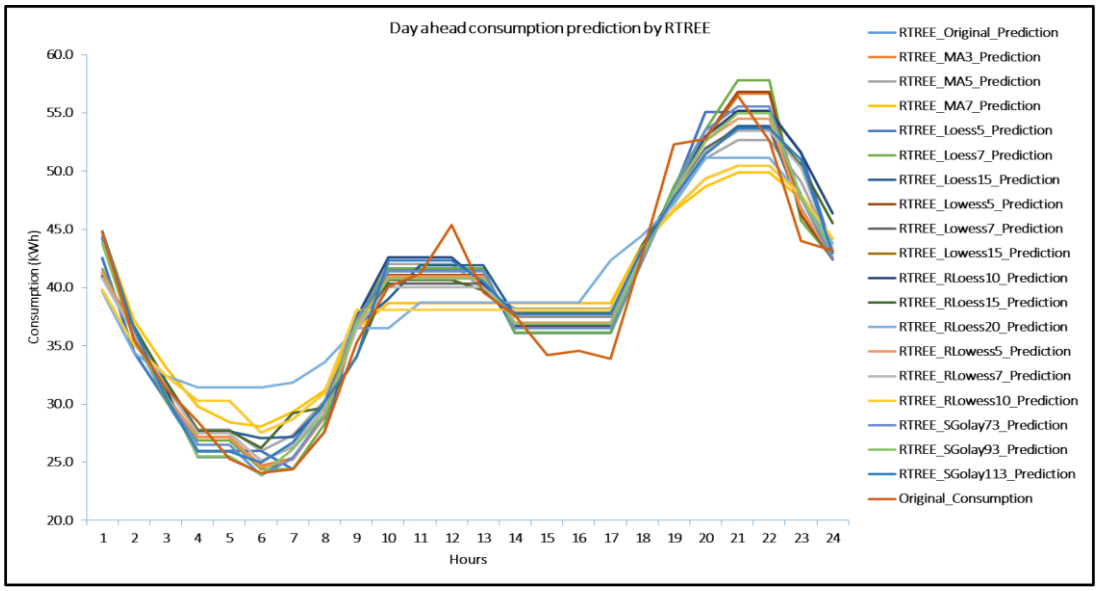


Figure 5.23: Day ahead electricity consumption prediction result by regression tree

Figure 5.24 shows the mean absolute percentage error for one day ahead consumption prediction using regression tree technique. The error is lower by about 1% from that of the neural network. The results are significantly improved from the neural network by a good margin. However the regression tree significantly dependent on the parameters and also on the amount of training data used. Besides, the regression tree also dependent on the underline distribution of the training data. If the distribution of training data is not uniform then the results yields from the regression tree are not good, however in our case the distribution is uniform and hence leads to better accuracy.

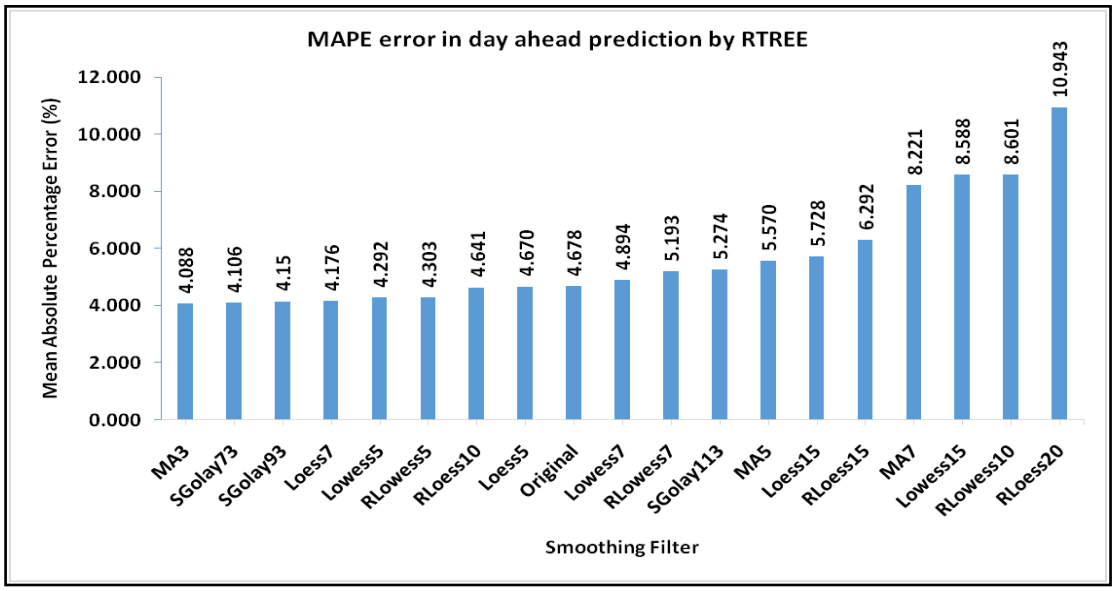


Figure 5.24: Mean absolute percentage error of day ahead prediction from regression tree

5.3.4. Genetic Programming in processing layer

Figure 5.25 illustrates the impact of pre-processing techniques on the results of the genetic programming based forecasting model. The results of best run of GP is shown on the pre-processed data as well as the smoothened data.

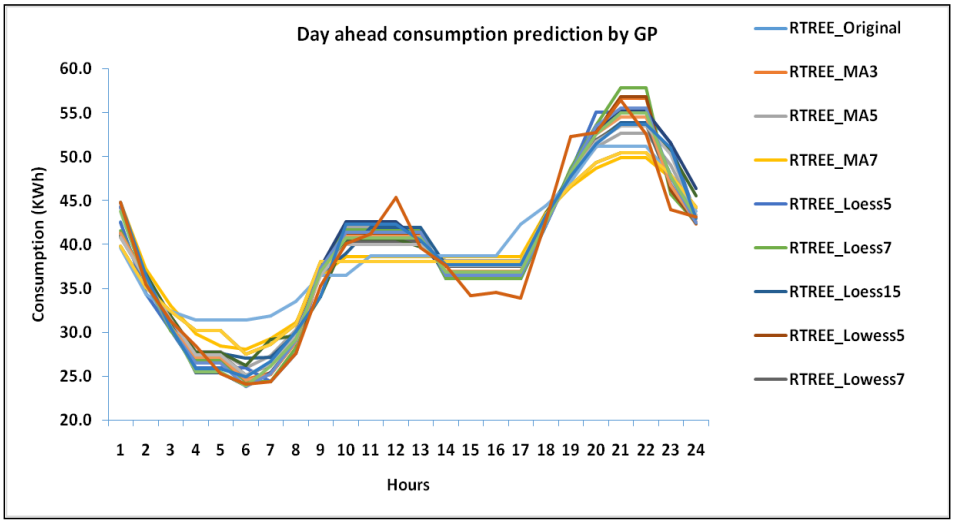


Figure 5.25: Day ahead electricity consumption prediction result by GP

Figure 5.26 shows the MAPE based error comparison of GP based consumption prediction. The error for different smoothing filters is sorted from lowest to highest MAPE error. Due to the fact that GP being an evolutionary approach does not produce the same results every time the same training data is presented. Therefore we calculated the best run, average run and worst run of GP run.

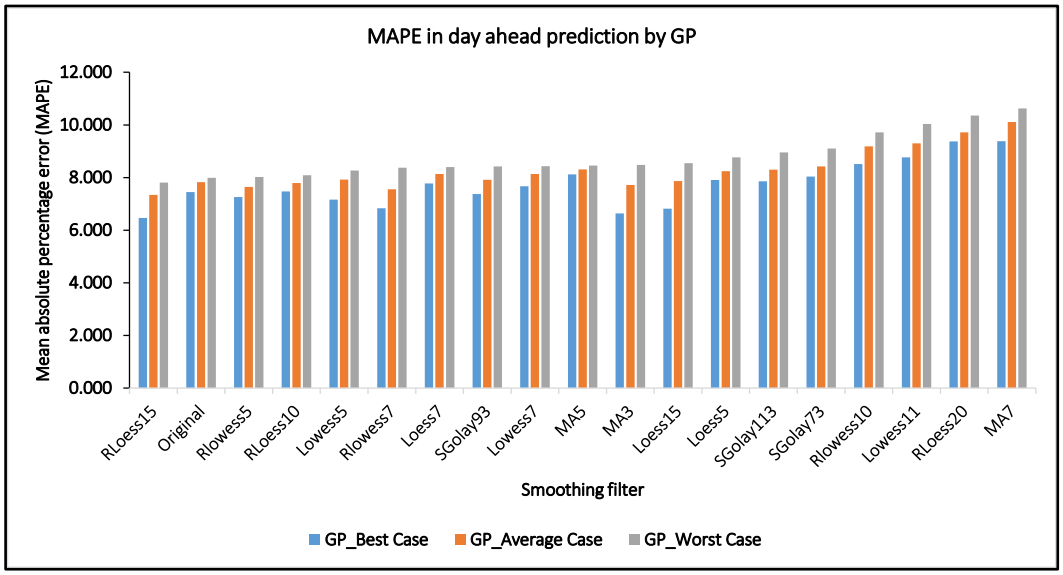


Figure 5.26 Mean absolute percentage error for day ahead consumption prediction using GP

Table 5.5: Mean absolute error values for day ahead prediction using GP

	MAPE		
Data	**GP_Best Case**	**GP_Average Case**	**GP_Worst Case**
RLoess15	6.468	7.342	7.810
Original	7.445	7.828	7.986
Rlowess5	7.260	7.644	8.018
RLoess10	7.471	7.795	8.084
Lowess5	7.157	7.924	8.263
Rlowess7	6.829	7.553	8.374

Loess7	7.776	8.137	8.401
SGolay93	7.370	7.912	8.420
Lowess7	7.665	8.139	8.428
MA5	8.119	8.309	8.458
MA3	6.636	7.719	8.482
Loess15	6.815	7.866	8.544
Loess5	7.909	8.245	8.764
SGolay113	7.861	8.302	8.953
SGolay73	8.035	8.421	9.104
Rlowess10	8.511	9.183	9.721
Lowess11	8.768	9.303	10.040
RLoess20	9.372	9.717	10.360
MA7	9.384	10.113	10.630

Table 5.5 shows the mean absolute percentage error of day ahead prediction from GP based prediction technique.

The Figure 5.27 shows the comparative results of the neural network, genetic programming and regression tree for a one day consumption prediction with a training set $T_{1.}$ The results shows that the regression tree (RTREE), outperforms the other counterparts. The margin is about 1% which means that the RTREE produces a much better results than the other two, as 1% is considered a significant improvement in MAPE in these scenario.

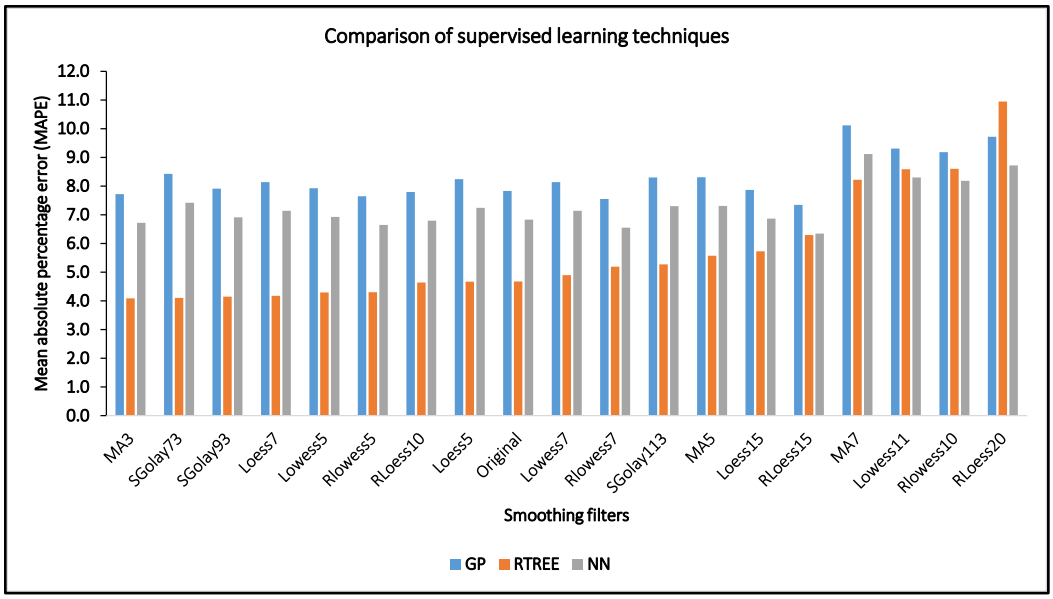


Figure 5.27: Comparison of MAPE for one day consumption prediction

5.3.5. Incremental learning based neural network in processing layer

Figure 5.28 shows the result of incremental learning based NN model. The training is carried for 720 hours and tested for next 10 days with incremental learning. The figure shows the result of 10th day without label data. The figure is very clear about the performance of the model which can extend the forecasting to the more days and long term forecasting.

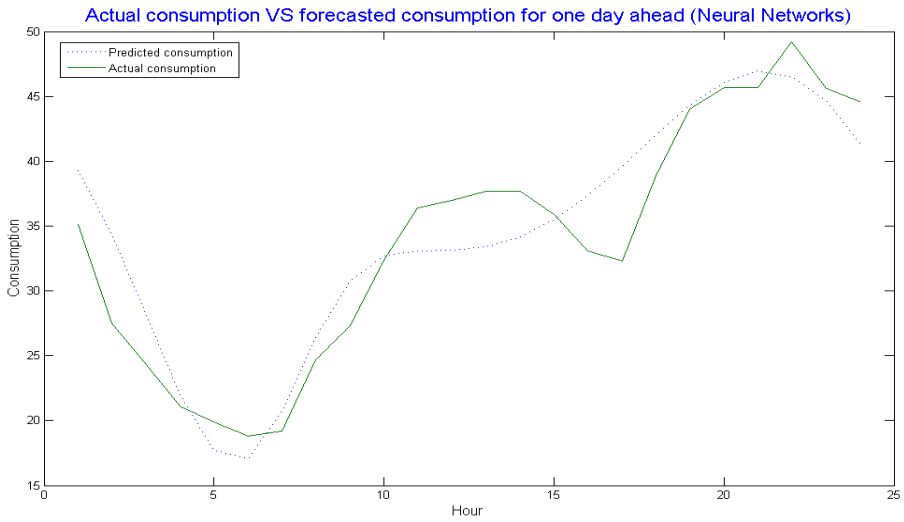


Figure 5.28: Incremental learning results

5.3.6. Ensemble Model in processing layer

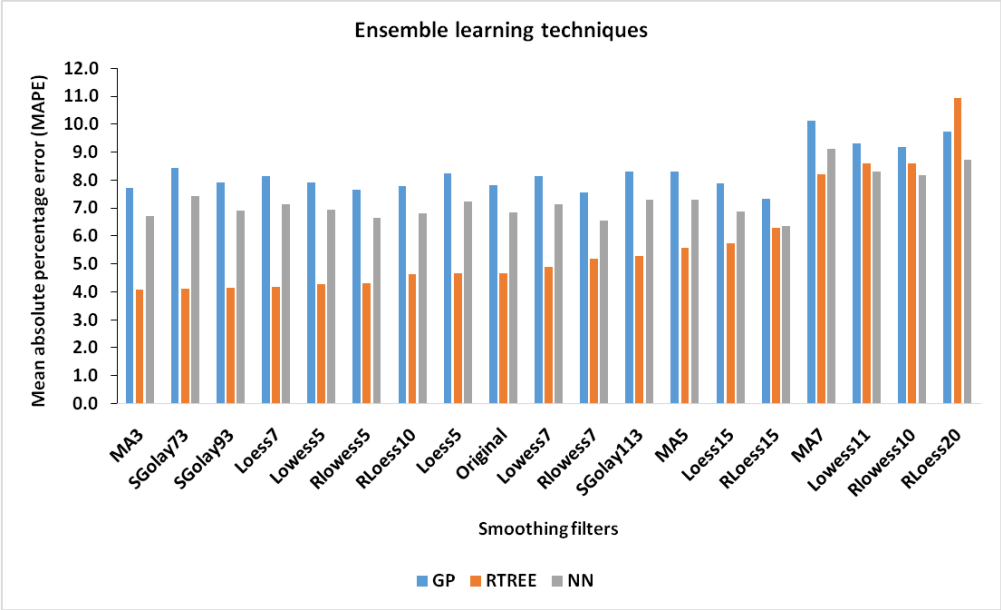


Figure 5.29: Ensemble model(Co-training) results

Figure 5.29 shows the results of hybrid approach through co training of RTREE, GP, and NN results. It is evident from the results that in the ensemble model RTREE remain the dominant

regressor in terms of the forecasting results. Therefore, RTREE is selected as decision maker, however in some cases NN perform better than RTREE so the decision is taken using the NN performance.

Table 5.6: Comparative results of supervised and semi-supervised approaches

Data	GP	RTREE	NN
MA3	7.719	4.088	6.72
SGolay73	8.421	4.106	7.42
SGolay93	7.912	4.15	6.91
Loess7	8.137	4.176	7.14
Lowess5	7.924	4.292	6.92
Rlowess5	7.644	4.303	6.64
RLoess10	7.795	4.641	6.80
Loess5	8.245	4.670	7.24
Original	7.828	4.678	6.83
Lowess7	8.139	4.894	7.14
Rlowess7	7.553	5.193	6.55
SGolay113	8.302	5.274	7.30
MA5	8.309	5.570	7.31
Loess15	7.866	5.728	6.87
RLoess15	7.342	6.292	6.34
MA7	10.113	8.221	9.11
Lowess11	9.303	8.588	8.30
Rlowess10	9.183	8.601	8.18
RLoess20	9.717	10.943	8.72

Conclusion: Due to the increase in electricity consumption in residential and commercial buildings, the proper analysis of electricity consumption deemed necessity. The electricity consumption analysis consist of electricity consumption patterns identification, the factors influencing the electricity consumption, the short term and long term prediction of electricity consumption etc. many studies exists for analyzing the electricity consumption in commercial buildings, however due to non-availability of data the residential buildings are not very explored.

In this work, the existing state of the art machine learning techniques are investigated and enhanced for electricity consumption in residential buildings. An enhanced incremental learning technique is devised for a long term electricity consumption forecasting. The prime focus of this work is on the short term electricity consumption forecasting in residential buildings. The data is from the urban area of South Korea. The results shows that the existing machine learning techniques well suited for the short term forecasting of the electricity consumption in residential apartments.

Furthermore, It has been observed from the experiments that human behavior is an important factor in predicting the electricity consumption in residential buildings at floor and apartment level. More extensive experiments for residential building from different urban and rural areas will help in building a more effective prediction model.

6. Experiment 3: Multimedia content recommendation in smart spaces

Home entertainment is a very important part of home life [91]. Home multimedia networks develop rapidly. People are paying more and more attention to services of information sharing, entertainment and communication within family members. Multimedia contents are an integral part of any multimedia service. With the advancement in technology, the internet get widen very rapidly. As the IoT has already been roaring around the research community, Cisco introduces a new terminology, called Internet of Everything (IoE) [92]. The IoE consist of people, process, data, and Things (Things from IoT). The idea to consider data as a thing brings multimedia contents to become part of IoT. Keeping in view of the IoE paradigm and TaaS (Thing as a Service), we evaluated our CIoT for a multimedia content recommendation case study in smart home.

Through multimedia home services, home residents can easily listen to music, radio and spoken information notifying the changes of home situation from smart home. Even residents move around different rooms. The CIoT services include autonomous adaptation according to user and environment context to carry our smart interactions based on personalized resident preferences.

Many approaches have been proposed from digital home to smart home considering the multimedia service deliveries. However, due to the size of multimedia contents, the heterogeneity of devices, and lack of standard framework the concept of automatic switching of multimedia contents is still lie in a dream land. A context-aware framework for uninterrupted provision of multimedia contents by integrating the concept of Internet of Things (IoT) and Service Oriented Architecture (SOA) has been integrated in the proposed CIoT. The objective

of this work is to improve the user's experience when using media resources in a smart home. Also, it has been considered that the use and access of media in any context and particularly in a resource-constrained context such as using mobile terminals through a wireless access. A home network is utilized for communication between the devices and proposed a device gateway to cope with devices heterogeneity. The simulation of the system shows that the proposed framework well suited for smart home and even scalable for larger scale.

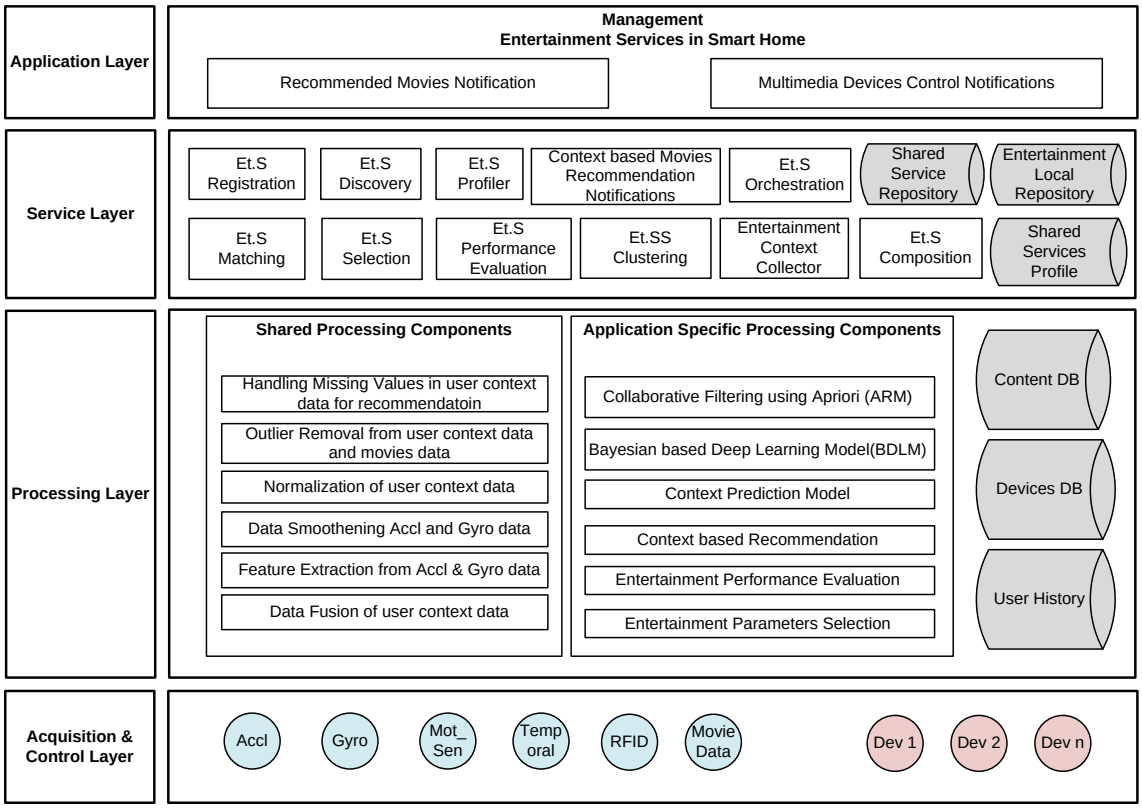


Figure 6.1: Layered component architecture of content recommendation

The layered component architecture is shown in Figure 6.1 which illustrates the component of the architecture used for the content recommendation purpose.

Application Layer: This layer is the consumer layer of the processing carried out at the lower layers. The application layer carry out all the interaction with the users, handle the

environment level details and proactively adapt the environment according to the user's context and the system's recommendation. As presented in Figure 6.1, two type of output is provided by this application i.e. multimedia devices control notification and recommended movies notifications. The devices control notifications are in the form of text messages while the recommendation are is in the form of listed recommendation with reliability index with respect to context.

Service Layer: This layer focuses on providing service as per the user preferences. User's location and time are considered as an abstract contextual information to know about the user activity. The 4 W's i.e. What, When, Where, Who are used as the detail contextual information in the proposed system. Following is the illustration of the 4 W's for our system:

- What: what happened (Activity)
- When: when happened (Time of the activity)
- Where: Where the activity carried out and what is the user location

Who: who performed the activity

The functionality of the components shown in Figure 6.1 is described in chapter 3, however, the specific detail of the components associated with this specific experiment is given below:

ServiceID	Name	Utilized by	Utilized at	Task accomplished	Used in consultation	Trust index with other services	Service consumed by Person ID	Service consumed at

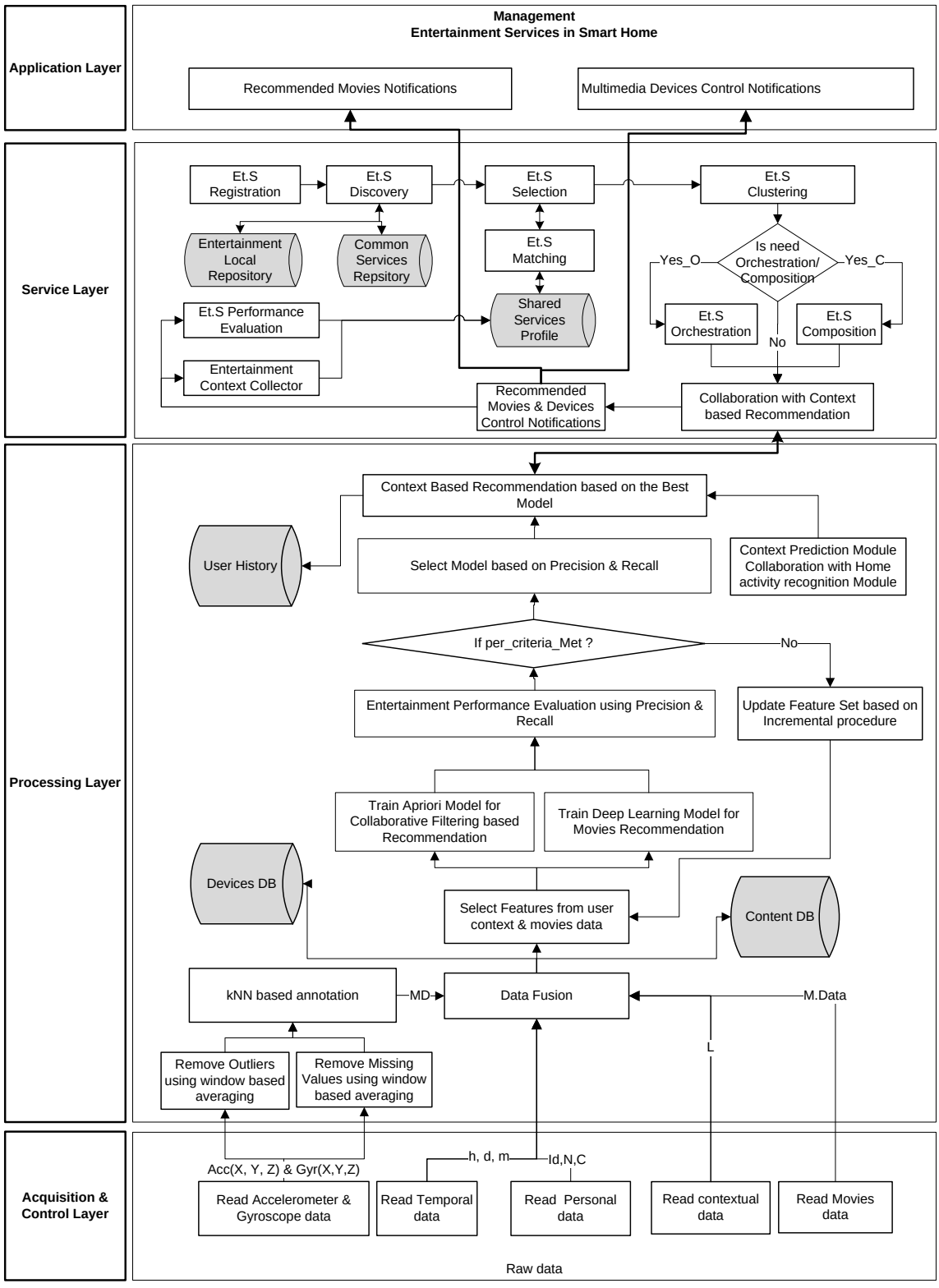


Figure 6.2: Flow model for entertainment content recommendation in smart home

Processing Layer: The processing layer consist of two major processing components i.e. deep learning based recommendation and apriori based recommendation of the multimedia contents. A detailed description of both the techniques is given below:

6.1. Deep Learning for content recommendation

Similar to the work presented in [71], the recommender system consider user rating [72] as the training and test data. The entire collection of I items (movies) is represented by a I-by-S matrix M_c, where row i is the bag-of-words vector $X_{c,i*}$ for item i based on a dictionary of size S. With U users, we define an U-by-I binary rating matrix $R = [R_{iu}]_{I \times U}$. For example, in the dataset "movie" & "netflex" $R_{iu} = 1$ if user u has history of i in his or her previous record and $R_{iu} = 0$ otherwise. The task is to recommend the content based on the predicted rating if a part of the ratings in R and the content information X_c is given. Although we focus on movie recommendation, the proposed our model is general enough to handle other recommendation tasks (article recommendation).

The matrix X_c is considered as noise-free matrix and is provided to the Stack Denoising Auto Encoders (SDAE) while X_0 is considered as a noisy matrix. The output of layer l of the SDAE is denoted by X_l which is a J-by-K_l matrix. Similar to X_c, row j of X_l is denoted by $X_{l,j*}$. W_l and b_l are the weight matrix and bias vector, respectively, of layer l, $W_{l,*n}$ denotes column n of W_l, and L is the number of layers. For convenience, we use W^+ to denote the collection of all layers of weight matrices and biases.

Bayesian SDAE: If we assume that both the clean input X_c and the corrupted input X_0 are observed, similar to [74]–[77], we can define the following generative process:

- ✓ For each layer l of the SDAE network,
- For each column n of the weight matrix W_l, draw

$W_{l,*n} \sim \mathrm{N}(0, \lambda_w^{-1} I_{Kl})$.

- Draw the bias vector $b_l \sim N(0, \lambda_w^{-1} I_{Kl})$.
- For each row j of X_l, draw

  $X_{c,j*} \sim N\left(\sigma\left(X_{l-1,j*}, W_l + b_l\right), \lambda_s^{-1} I K_l\right)$.

✓ For each item j, draw clean input

$$X_{c,j*} \sim N\left(X_{L,j*}, \lambda_n^{-1} I_J\right).$$

Deep Learning Model (DLM): We followed the exact model presented in [78] and Bayesian SDAE as our base model for the content recommendation task. The steps followed are as follows:

1. For each layer l of the SDAE network,

   (a) For each column n of the weight matrix W_l, draw

$$W_{l,*n} \sim \mathrm{N}(0, \lambda_w^{-1} I_{Kl})$$

   (b) Draw the bias vector $b_l \sim N(0, \lambda_w^{-1} I_{Kl})$.

   (c) For each row *j* of X_l, draw

$$X_{l,j*} \sim N\left(\sigma\left(X_{l-1,j*}, W_l + b_l\right), \lambda_s^{-1} I K_l\right).$$

2. For each item j,

   (a) Draw a clean input $X_{c,j*} \sim N\left(X_{L,j*}, \lambda_n^{-1} I_J\right)$.

   (b) Draw a latent item offset vector $\epsilon_j \sim N(0, \lambda_v^{-1} I_K)$ and then set the latent item vector to be:

$$v_j = \epsilon_j + X^T_{\frac{L}{2},j*}$$

3. Draw a latent user vector for each user *i*:

$$u_i \sim N(0, \lambda_u^{-1} I_K)$$

4. Draw a rating R_{ij} for each user-item pair (i, j):

$$R_{ij} \sim N(u_i^T v_j, C_{ij}^{-1})$$

Here λ_w, λ_n, λ_u, λ_s, λ_v are hyper-parameters and C_{ij} is a confidence parameter. The middle layer $X_{L/2}$ serves as a bridge between the ratings and content information. This middle layer, along with the latent offset ϵ_j, is the key that enables DLM to simultaneously learn an effective feature representation and capture the similarity and (implicit) relationship between items (and users). Similar to the generalized SDAE, for computational efficiency, we can also take λ_s to infinity.

Prediction with DLM: Let D be the observed test data. Similar to [34], we use the point estimates of u_i, W^+ and ϵ_j to calculate the predicted rating:

$$R_{ij}^* \approx (u_j^*)^T (f_e(X_{0,j*}, W^{+*})^T + \epsilon_j^*) = (u_i^*)^T v_j^*$$

Note that for any new item *j* with no rating in the training data, its offset ϵ_j will be 0.

6.2. Apriori based content recommendation

The Apriori algorithm work as scanning the dataset every time it generates the candidate item sets to build the association rules [5]. Our proposed mechanism apply the Apriori algorithm on two dimensional spaces (*User Item (movie)*) with many users and fewer number of items. Moreover, most current recommendation systems consider the items that have been rated highly by the users and recommend similar items to the target user. The existing recommendation systems focuses on items/contents that are liked or explicitly feed backed by the user. The

contents/items that does not have explicit feedback by the user is normally discarded for that specific user. The assumption in our work here is: Even if user do not have an explicit feed back or didn't liked an item (action movie) that she/ he watched, the system should be smart enough to recommend another action movie to the user.

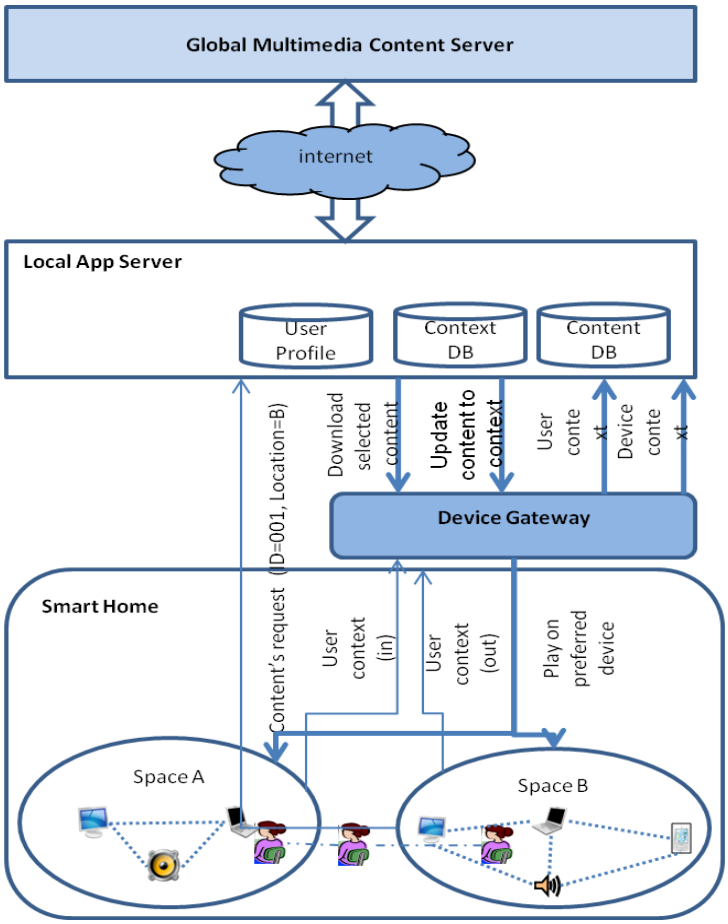


Figure 6.3 Conceptual model for intelligent multimedia services

The architecture of the system which utilizes the Apriori based recommendation is shown in Figure 6.3. The system's backbone is the local app server which collects the user information, find the user context and take decision for service delivery. The services are initialized on the local app server, device gateway, and global content server. User make request for the contents and devices to the local app server. The local app server is further connected with global content

server and device gateway. The local app server keeps the contents which are downloaded from the global content server on user demands. The global content server is supposed to have all the contents which the user request. Besides, local app server also keeps the necessary information like, type of contents, category, genre, release date, subtitles, ratings, and language.

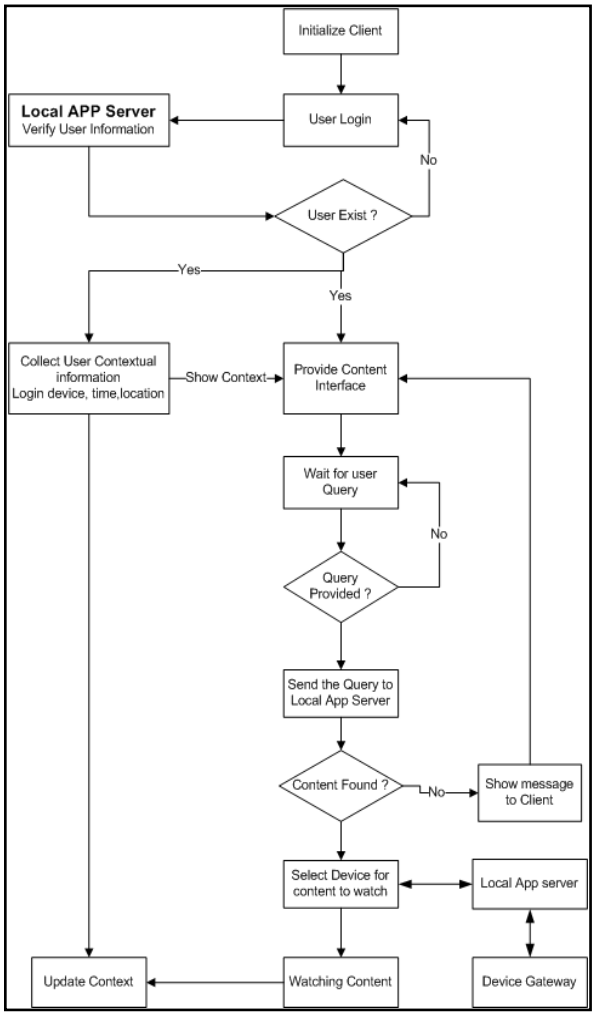


Figure 6.4: Flow diagram at the client application

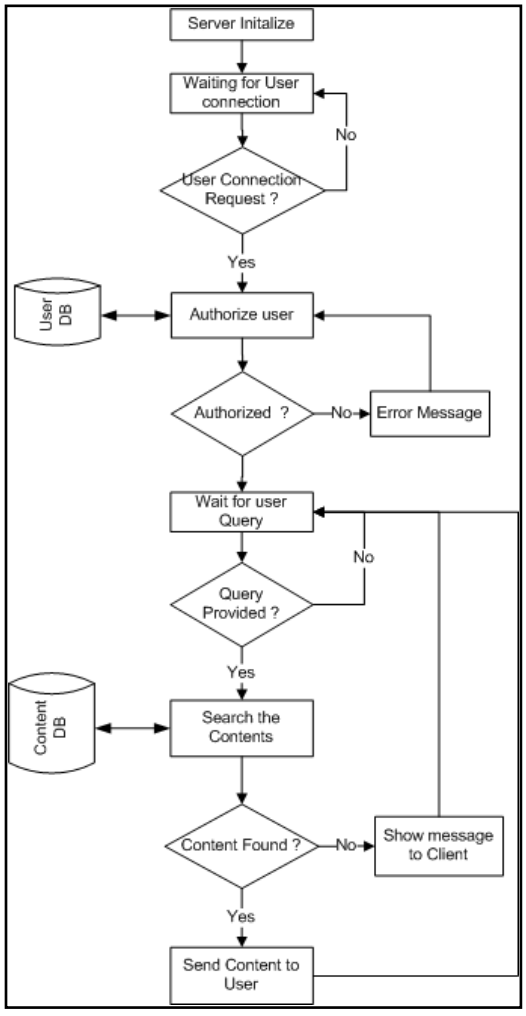


Figure 6.5: Flow diagram for local app server

The modules of the architecture are described in the following sections.

- **Users:** The basic role of client part in our system is content querying and viewing. The user of the system must be registered with the system to use the services. Figure 6.4

shows the flow chart at the client application. The user query for the grant of permission to the system and local app server authenticate the query if the user is already registered. The user then put the query for the contents and can watch the multimedia contents on the registered devices. The content query consists of content name or content type, year of release, or category. When the user generates the query for the content to watch/listen, it is forwarded to the local app server. The local app server checks the context of the requested contents, devices and user and recommends the best combination for the user. However, the user has to decide for the content and devices. The local app server sends the required information to the client and then client selects the required contents and device. The client will be shown with the required contents on the selected device.

- **Local App Server (LAS):** The local app server is the most important part of our system. The local app server is the content base, context base and query handler for the contents. Also, the LAS communicate with the global content server to update the local database for the newly released multimedia contents. The local app server also collects the contextual information of the home, users, devices, and contents. The context reasoning is carried out at the LAS to infer the user's activities and status for multimedia contents. To the best of our knowledge it is a novel entity in the smart home scenario. When a request is received from the client for a device, the app server executes the algorithm 1 to find the status of the devices. The server request the device gateway for the devices status, the device gateway responds in the devices status. The local app server differentiates the busy and free devices based on their status. Server issue a command to that device through the device gateway. The device status is updated in the database of the gateway and also the app server keeps track of the device. The device information is also displayed to the user on which he/she is watching the contents. Besides the device

status is received at run time from the device gateway. The device gateway can turn on and turn off any device. Figure 6.5 shows the flow chart for the local app server. It is evident from the figure that when a request is received from the client for a device, the app server checks the status of the devices. The server request the device gateway for the devices status, the device gateway responds in the devices status. The local app server differentiates the busy and free devices based on their status. The app server sends the user with all devices as free and busy devices. The user can select the devices from the lists. If the user select a busy device, then the user is warned either to stop the already running device and also the system check the authorization of that user to stop the device. There are two scenarios for the user to select the device i.e. a free device and a busy device. When a free device is selected, the app server considers only the authorization level of the user. If the authorization level permits the access of that device to the user then the app server issue a command to that device through the device gateway. The device status is updated in the database of the gateway and also the app server keeps track of the device. The device information is also displayed to the user on which he/she is watching the contents. Besides the device status is received at run time from the device gateway. The device gateway can turn on and turn off any device. The basic information along with the address of the content is stored in the local app server's database. The content is then transferred to the selected/preferred devices. Contextual information plays a vital role for providing smart personalized services. The local app server collects the contextual information about the user's location from the motion sensors installed in the smart home, the time of day and day of week. This information is considered as a context pattern for the provision of smart services.

a) User ID

b) Profile

c) Location

d) Time of day

e) Day of week

f) Working day (yes/no)

The content's context is provided by the content DB at the local app server, the user history maintained at the local app server and the device's context is taken from the device gateway which keeps updates about all the devices in its range. Besides, the existence of a user or his activity is also considered as contextual information for smooth services. Once the context data is collected, the local app server makes decision regarding the service providing to the user. The location sensors always keep track of the user's location, when the user changes the location, the system identify the user's activity. If the activity is identified as a long time activity the local app server through the device gateway pauses the content play on the device and takes according action.

Algorithm 2 is used for finding contents in the app server. When user make request for the contents, the procedure that local app server adapts is listed in algorithm 2. There are two procedures for contents search i.e. first; all contents of the local app server, second; search a specific content with the name. If the user requests all the contents, then the local app server sends the list of all the contents to the user. The client application is enhanced to filter the contents based on the type, category or release date. In the second case, if the user requests for a specific contents, then the local app server search the its own database, if the contents found, it is transferred to the requested device, but if the contents does not exists with the local app server then the local app request the global content server and downloads the contents. The basic information along with the address of the content is stored in the local app server's database. The content is then transferred to the selected/preferred devices.

Algorithm 1: DeviceStatus (device_query)	**Algorithm 2: SearchContent()**
Input: Client Request for devices **Output:** Free devices Busy devices **Procedure Devices()** { *Initialization* *FList=0; // free devices* *BList=0; // busy devices* *S=null;* *DList = DeviceList();* *Count = DList.count(); //count devices* ***for** i ← 1 to Count **Do*** *{* *S=DeviceStatus(i); // from device gateway* *If S==Free then* *Y.add(i);* *else* *X.add(i);* *}* *return DList, X, Y;* }	**Input:** Client Request for contents **Output:** ContentsList **Procedure AllContents()** { *Initialization* *DBConn=DatabaseConnection();* *Using DBConn search the database for contents* *LClist=ContentList(); //Local Contents* *return LClist;* } Procedure FindContents(name) { *Initialization* *Content=null;* *DBConn=DatabaseConnection();* *Using DBConn search the database for contents* *LClist=ContentList();* *Count=LClist.count();* ***for** i ← 1 to Count **Do*** *{ If(LClist[i].name=name)* *{ Content=LClist[i]; }* *}* *If(Content=null)* *{* *Request Global Server for content* *Content = GlobalServer(name);* *}* *return Content:* }

The Local App server work as context base and work on our context model. The contextual information model is illustrated in Figure 6.6. As shown in figure 3, the local app server sends available devices and available contents to the user query. The local app server makes this decision based on the available contextual information in the local

app server. We consider the device functional status (functional/non-functional), available contents, content's reviews and content's subtitles status as context information. These parameters are also used as contextual information for the device recommendation. The history under the local app server is updated based on the user content selection.

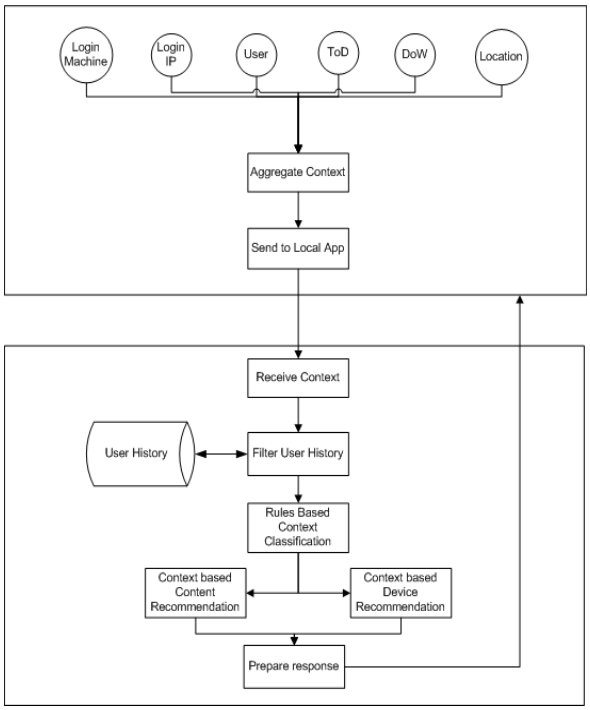


Figure 6.6: Flow of context processing at the local app server

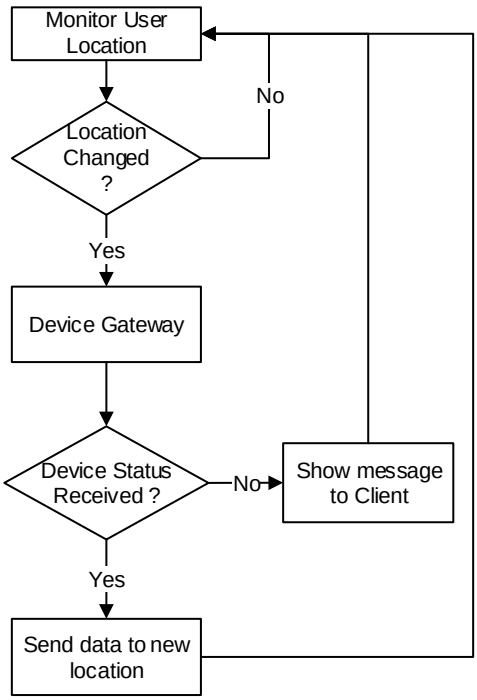


Figure 6.7: User location service

- **Global Content server:** The Global content server has a very simple role in the system. The global content server is supposed to have all the contents that are required to the user of the system. The local app server has the access to this server. The user requests based on the contents name, is handled by the global contents server. The global content server searches the requested content in the database. When the contents finds it sends the basic information of the contents to the local app server. When the client request to

start watching a content which is available on the global content server, the user is informed to wait and the local app server request the global contents server for download of the contents. When the download completes at the local app server, the content is directed to the selected device of the user.

- **Device Gateway:** The device gateway works as a middle ware between the devices and the local app server. The device gateway keeps track of the devices usage and their status. When the local app server requests the device gateway to start or stop a device, the gateway first checks the status of the device and send a response to the local app server. The gateway also keeps track of the functional status of the devices. The gateway also has the facility to register a new device or delete any device.

Sequence diagram

The Figure 6.8 shows the flow chart of our system. The local app server have six objects namely, user, Local App Server, Global Contents Server, Device Gateway, and Devices. The flow diagram shows the lifeline and an execution of a request from the user to the devices and response to the user. The request initiated by the user results in a show of contents to user's selected device. The context sharing mechanism is also shown in the figure.

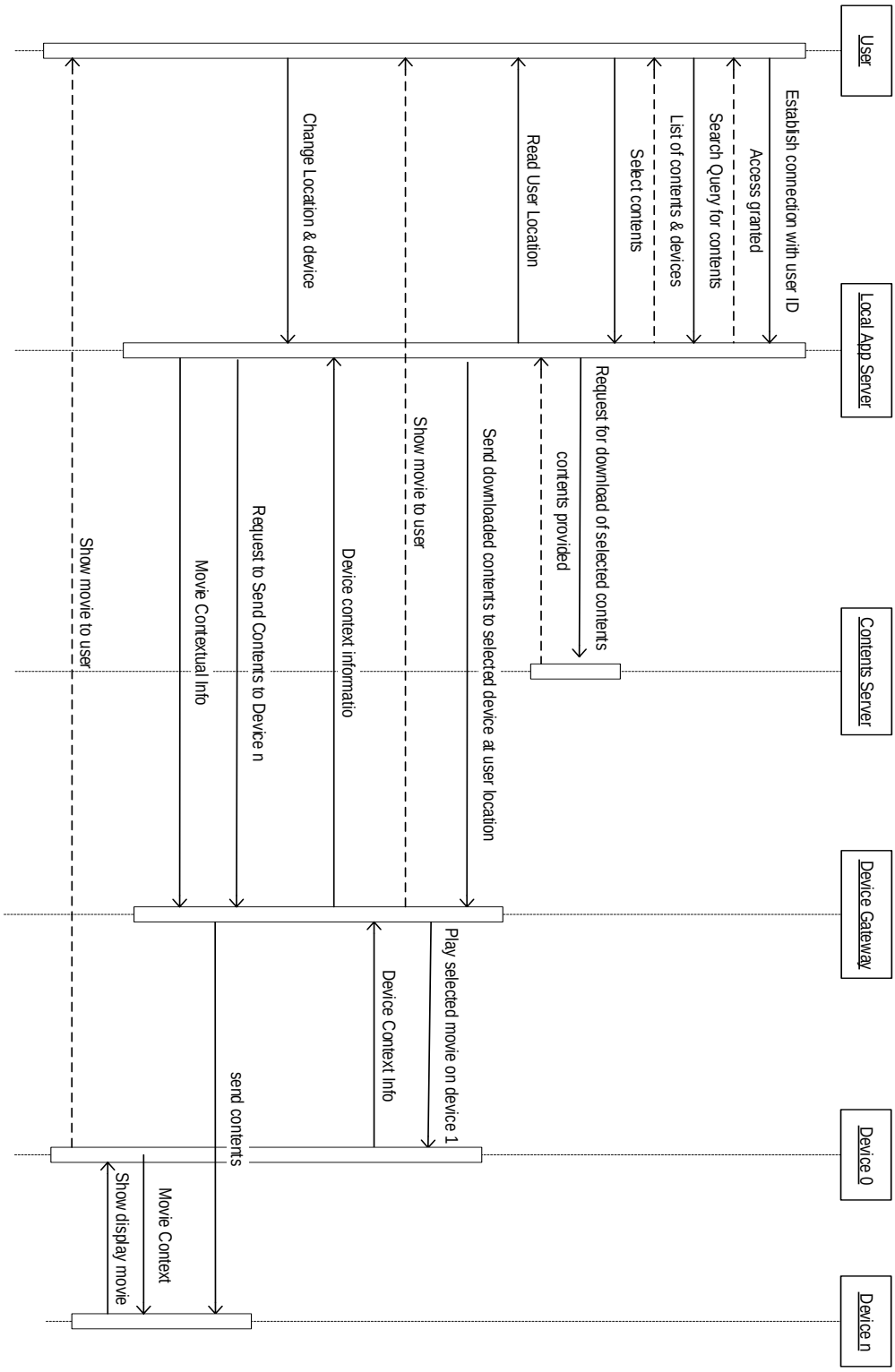


Figure 6.8: Sequence diagram for intelligent multimedia services

Business Process Modeling

- **Users:** The basic role of user pool as shown in the Figure 6.9 is to select contents and watch on the preferred devices. The user must be registered in the system to get smart personalized services, therefore the user login with the id and password and make the content query which consists of content name, content type, year of release, or category. When the user generates the query for the content to watch/listen, it is forwarded to the local app server. The local app server sends the requested information along with the recommended contents based on the use history, and then client selects the required contents and device. The client watches the selected contents on the preferred device. The system sense the user location and provide the content based on his activities and location.
- **Local App server:** The local app server is the most important part of our system. When the user generates the query for the content to watch/listen, it is forwarded to the local app server. The local app server sends the required information to the client and then client selects the required contents and device. Besides, the local app server keep history (content watched, time of day, day of week, device used etc.) of the user's activities regarding the multimedia services in the home. Utilizing the history information of the user, the system build a recommender system which recommends the contents to the user along with the preferred devices. The user preferences for devices are taken into account for prioritizing the device in a multi-devices location. Furthermore, local app server also work as a bridge between the user and global content server, which means that the queries for content to global content server will be made by local app server not by the user, thus the global content server will not be directly accessible to the user and hence will lead to a secure environment.

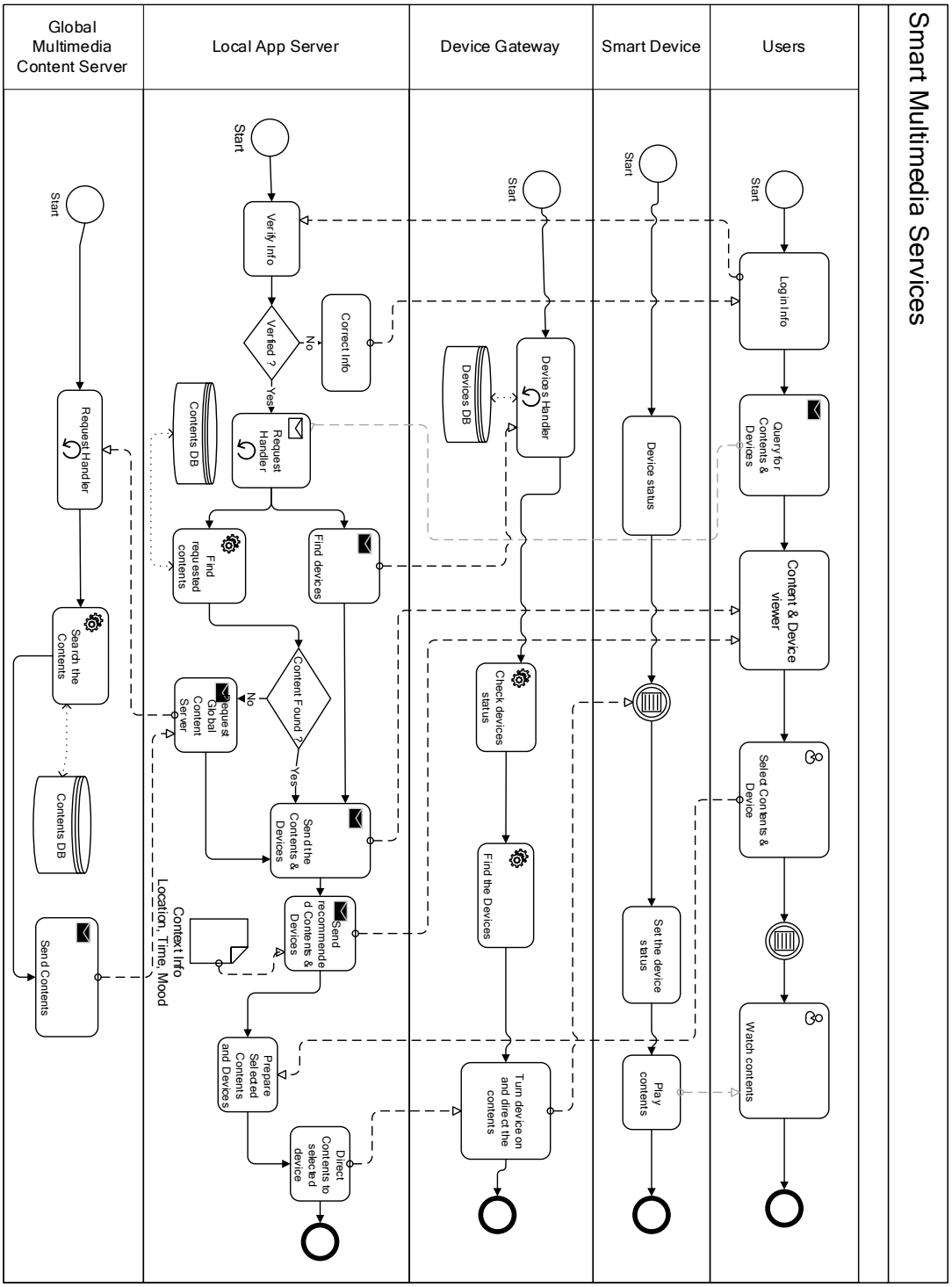


Figure 6.9: BPM for Smart Service provisioning

- **Global Content server:** The Global content server has a very simple role in the system. The global content server is supposed to have all the contents that are required to the user of the system. The local app server has the access to this server. The user requests based on the contents name, is handled by the global contents server. The global content server searches the requested content in the database. When the contents finds it sends the basic information of the contents to the local app server.
- **Device Gateway:** The device gateway works as a middle ware between the devices and the local app server. The device gateway keeps track of the devices usage and their status. When the local app server requests the device gateway to start or stop a device, the gateway first checks the status of the device and send a response to the local app server.

Conclusion: In this work, we propose a context aware content sharing for smart home. Through this approach home residents can enjoy ubiquitous multimedia services in home. Using context ontology, it can assist the home reasoning over low-level raw contexts to derive out the high-level abstracting context. Thus our system can easily apply the context aware concept to perform feasible multimedia service for home residents based on the inferring result of collected context by home. Although the proposed system can provide various audio-based multimedia services for home residents in smart home, there are still several new trends and issues of smart home that can be integrated into our system, such video-based multimedia services, advanced identifying and tracking mechanisms – bioinformatics identification and wireless sensor networks, home automation, environmental surveillance and home security, home healthcare, etc. Through ubiquitous and context aware computing technologies to achieve above services, we believe that the dream of smart home will come true in the future.

6.3. Data set and experimental configuration

For evaluation of our technique, all computations were performed on AMD Athlon (tm) 64-bit X2 Dual Core 6000+ 3.10 GHz processor with 4GB Physical Memory. However, the program was built using 32-bit compiler, which means full computational power is not utilized. Table 6.1 shows the implementation environment of our content sharing service. it shows the technologies used for different level of implementation.

Table 6.1: Environmental configuration for the implementation of the multimedia services

S.No	Name of Object	Programming Language	Technology
1	Client App	C#	• Visual Studio 2013, • .Net framework 4.0
2	Device Gateway	C#.Net, WCF Services, Sql Server	• Visual Studio 2013, • .Net framework 4.0 • Sql Server 2012
3	Local App Server	C#.Net, WCF Services, Sql Server	• Visual Studio 2013, • .Net framework 4.0 • Sql Server 2012
4	Global Content Server	C#.Net, WCF Services, Sql Server	• Visual Studio 2013, • .Net framework 4.0 • Sql Server 2012

Figure 6.10 shows the user interface of the local app server. The local app server user interface consists of start multimedia service option for the local app services, register new contents and a view of the active sessions and the services sessions and the current context of the system. The local app server also keeps track of the connected user with the user name, the IP, and the device type of the user. The administrator of the system can see the active sessions and the communication load on the local app server. The figure also illustrates the initialization of the content service. The local app server shows the following information:

- Available contents

- Active session
- Service status
- Current context

The active session enlists the user and devices information which are currently using multimedia contents.

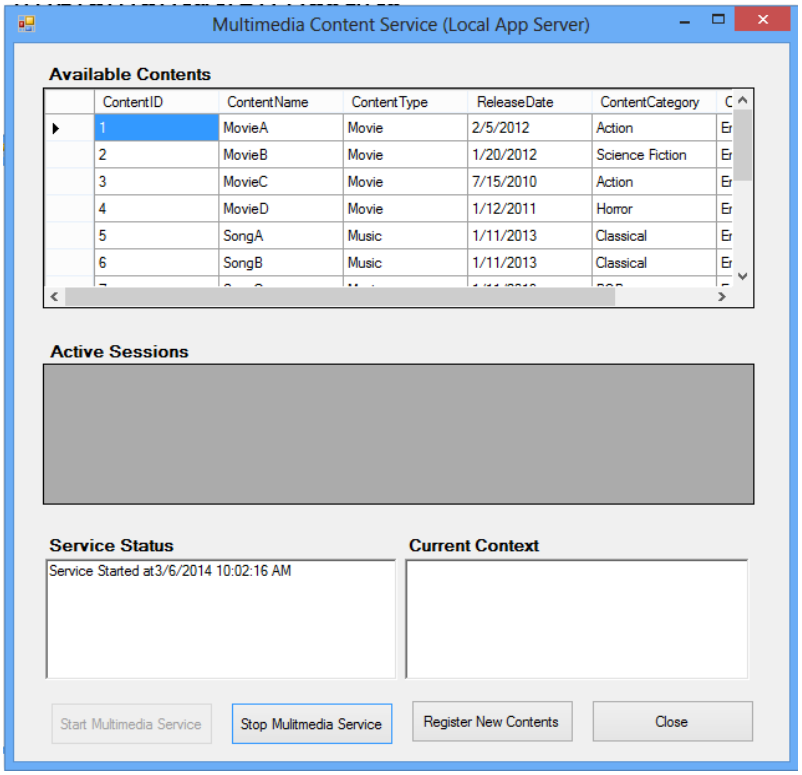


Figure 6.10: Local App server user interface

The content registration form is shown in Figure 6.11, depicting the data required for the new content registration. The content registered through the content registration form is stored in the database at the local app server. The system administrator are allowed to register new contents and the client can only view the content.

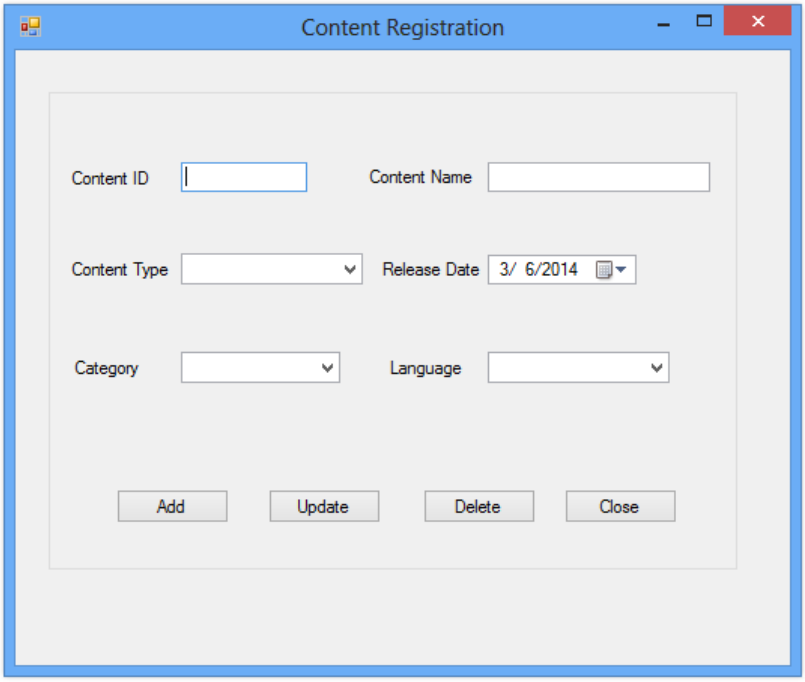


Figure 6.11: Content registration interface

Device gateway interface is shown in Figure 6.12, which depicts the service log, device context, device service start and stop, and device management service. The services log shows the service utilization log and devices context shows the current status of the devices. The device management functionality of the device gateway handles the devices addition, deletion and updation of the devices as shown in Figure 6.13.

Figure 6.12: Devices gateway interface

The Figure 6.13 depicts the devices information along with the additional functionality of addition of new devices and checking the status of a specific device. The new devices registration will required the device id, device name, type, status, functional status and location. This data is later updated by the recommender system as the context of the devices changes.

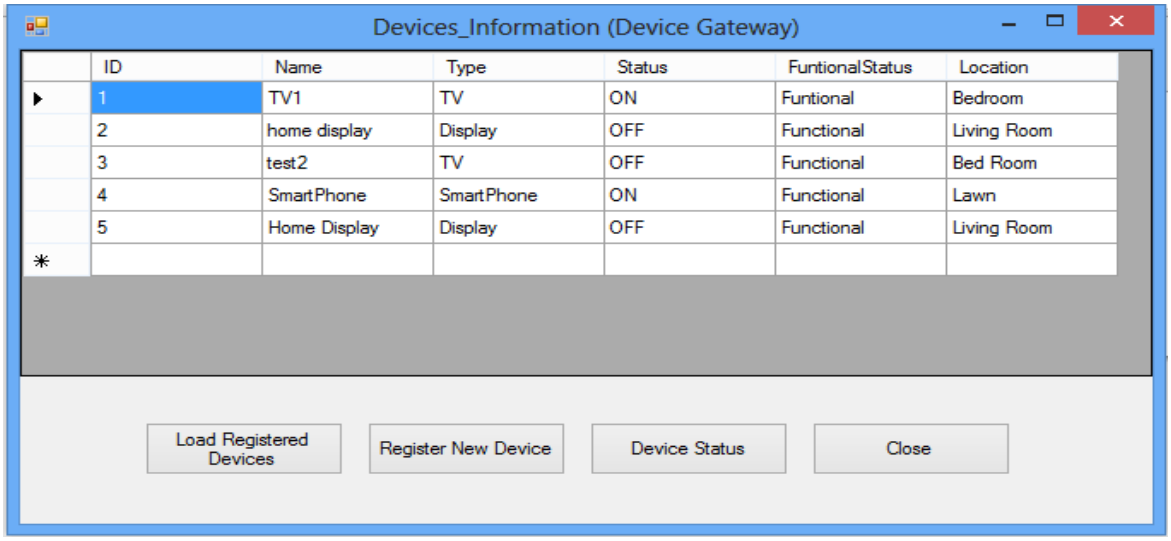


Figure 6.13: Devices management at device gateway

Figure 6.14 shows the user interface of the device registration. The new devices are registered through the form shown in this figure. The devices are registered at the devices gateway which further share the information with the local app server.

Figure 6.15 shows the smart television emulator, showing the basic information of the television, contextual information and the status of the device. This emulator is controlled by the device gateway by getting the recommended notifications from the local app server.

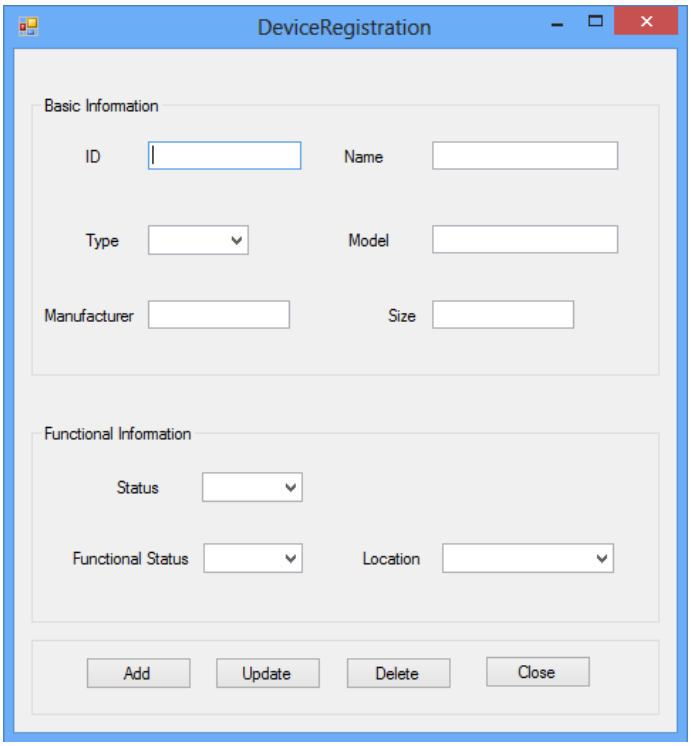


Figure 6.14: Device registration interface

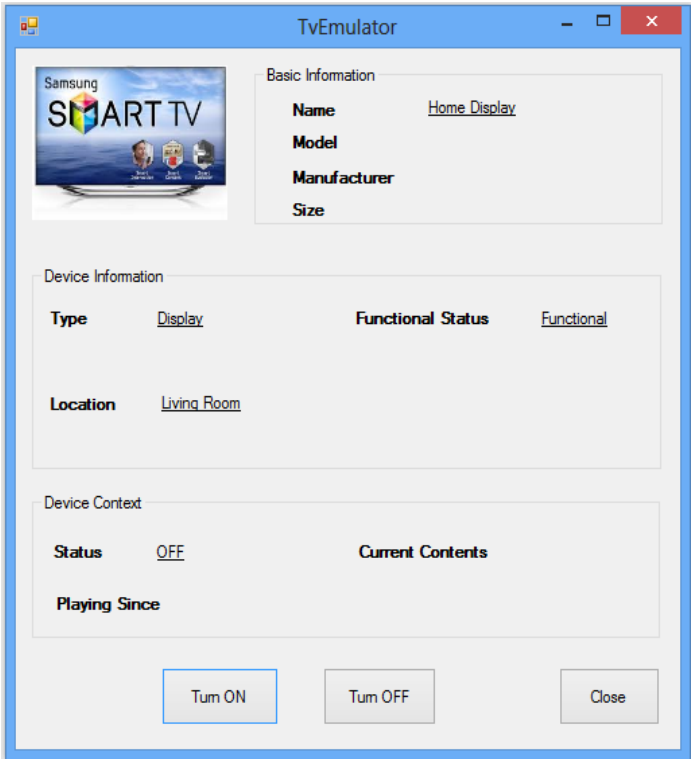


Figure 6.15: Smart television emulator

The Figure 6.16 shows the user interface (UI) of the client unit of the system. The client UI consists of the query section and the content viewer section. Also the client UI consists of device and content selection section where the users view the lists of contents and devices. Besides, it also shows the device and IP information to the user.

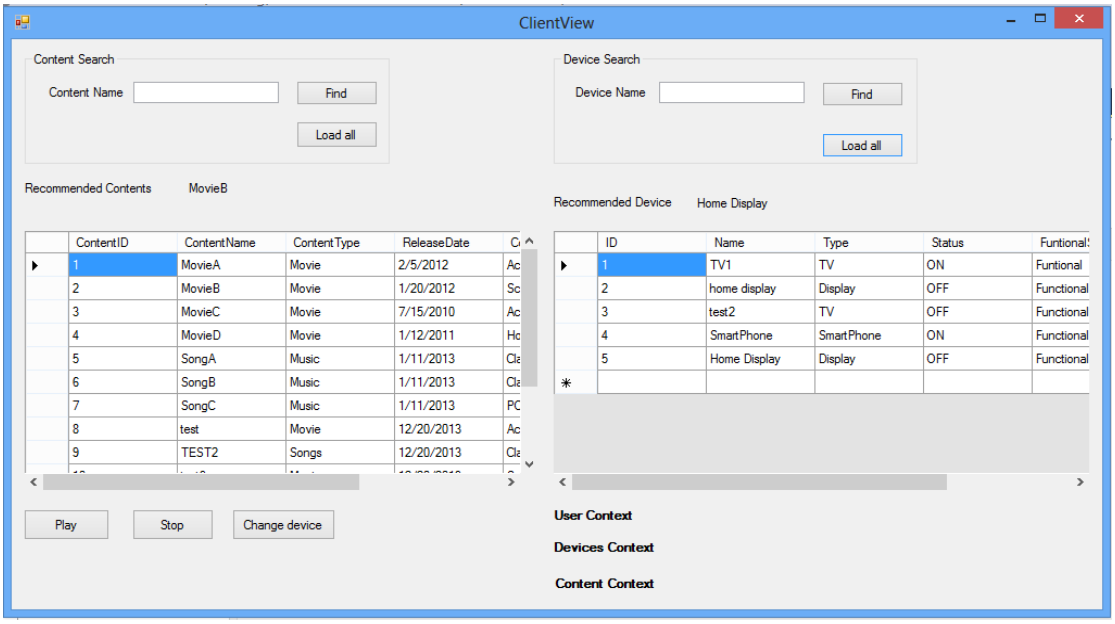


Figure 6.16: Client user interface

Figure 6.17 shows the client view of the registered devices and registered contents. The user put the query for the contents and devices. The LAS respond in the contents and devices with the recommended content and devices as per the user context. However, the data shown in the figure is just a randomly filled data in the database. Therefore the names and ids are not consistent.

Figure 6.18 shows the user interface for the global content service. The interface consist of service initialize and service terminator option, which means the service can either be started or stopped from this user interface. User Interface (UI) of the global content server shows the log of the service, which keeps track of the service start and stop timing. This helps in determining the service execution time and profiling of the service.

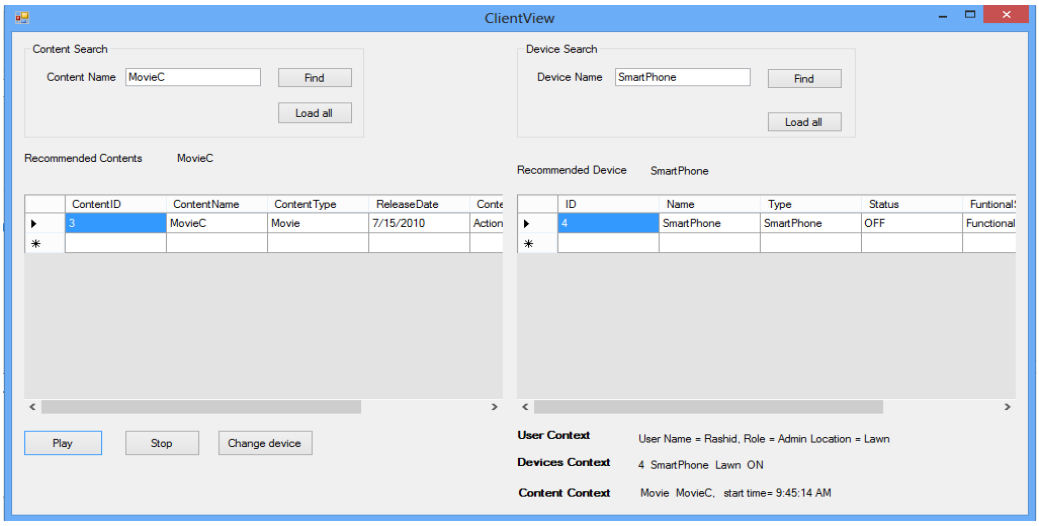


Figure 6.17: Client UI for contents and devices view

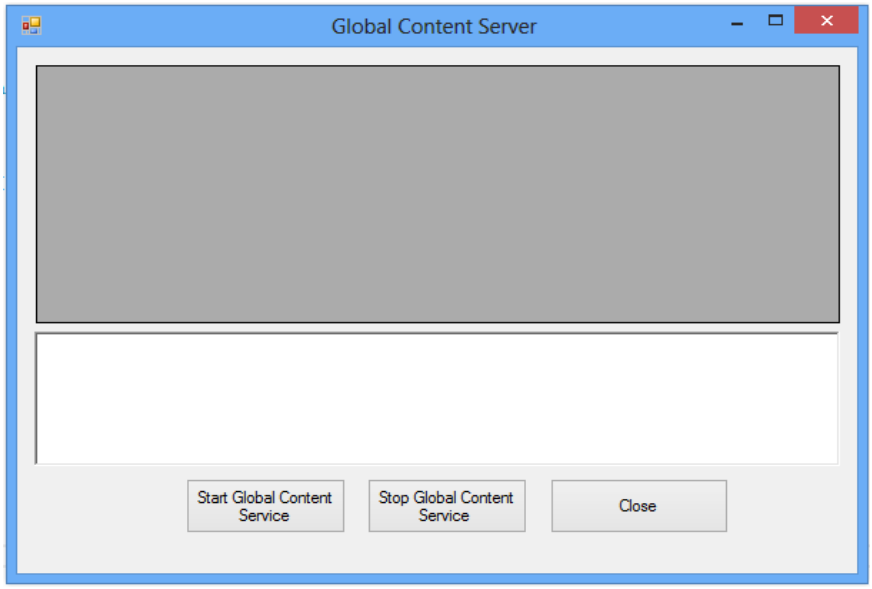


Figure 6.18: User interface of the global content server

6.4. Evaluation mechanism

In order to measure the closeness of predictions to users' real preferences, a numerical representation is normally used. In addition, for reasons of clarity, we will use the same notation along the current section. To this end, let us call P(u, i) the predictions of a recommender system

for every particular user u and item i, and p(u, i) the real preferences. Clearly, the function p(u, i) can never be known with absolute precision. Therefore, the values of this function are most usually estimated by means of the users' previous ratings. As we said above, these ratings can be obtained explicitly or implicitly.

In some cases, both functions p(u, i) and P(u, i) will offer only two values 1 or 0, which means that a particular item I is considered useful or useless, respectively, for a particular user u. For this singular case, we will say that p and P are binary functions.

Accuracy metrics: Accuracy metrics measure the quality of nearness to the truth or the true value achieved by a system. Perhaps, accuracy is the most used and well-known metric into the field of Artificial Intelligence. Particularizing the metric to the recommender system's field, it can be formulated as shown:

$$accuracy = \frac{number\ \ of\ good\ \ cases}{numbe\ r\ of\ \ cases}$$

$$accuracy = \frac{number\ \ of\ successful\ \ recommendation}{number\ \ of\ recommendations}$$

Now, assuming that a ''successful recommendation'' is equivalent to ''the usefulness of the recommended object is close to the user's real preferences'', and using the functions *p* and *P* introduced previously, we may be more formal and reformulate accuracy as shown below. In this equation, we consider that *p* and *P* are binary functions. Additionally, *r(u, i)* is 1 if the recommender showed the item *i* to the user *u*, and 0 otherwise. Finally$R = \sum_{u,i} r(u,i)$is the number ofrecommended items shown to the users.

$$accuracy = \frac{\sum_{(\forall u,i/r(u,i)=1)} 1-|p(u,i)-P(u,i)|}{R}$$

Also common in the recommender systems' field is the metric mean absolute error (MAE). This metric measures the average absolute deviation between each predicted rating *P(u, i)* and each

user's real ones *p(u, i).* Then, due to the fact that only rated items can show us each user's real preferences, we may derive the (6), where i must have been rated by u (to obtain p(u, i)). In this, we consider N as the number of observations available, which obviously depends on the number of items properly rated. Of course, the higher this number, the better the estimate.

$$MAE = \frac{\sum_{u,i}|p(u,i)-P(u,i)|}{N}$$

6.5. Results and discussion

The preliminary results are recorded for the proposed system. We produced a random dataset of audios and videos. The data sets were divided into five categories i.e. dataset1, dataset2, dataset3, dataset4, dataset5.

6.5.1. Multimedia content recommendation results

The Figure 6.19 shows the results of all the datasets including the combination of all the datasets. Due to the reason that the movies sets are randomly recorded, therefore the accuracy decreases with the number of user increased.

old Cross alidation	*MAE*	*RMSE*
1st	0.669324874	0.871004639
2nd	0.691398246	0.869712575
3rd	0.815557926	1.037908492
4th	0.559849015	0.677475326
5th	0.696621347	0.93701284
Mean	0.6865502804	0.8786227744

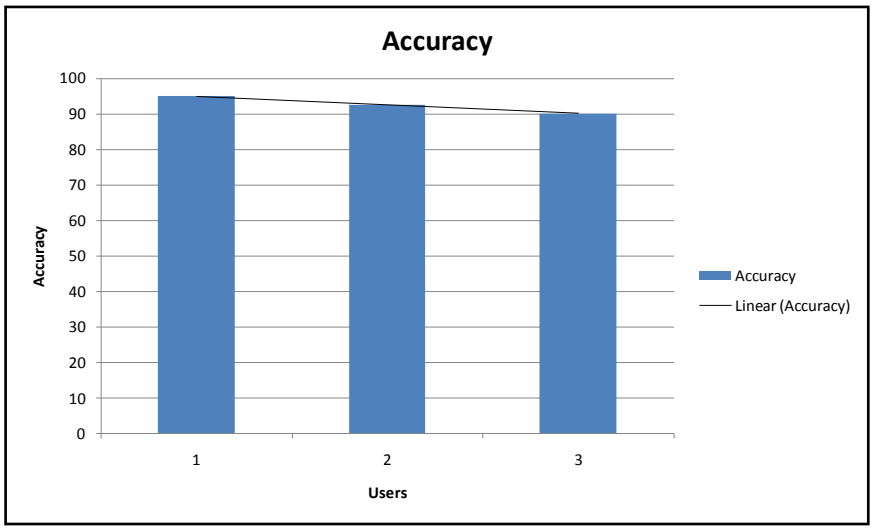


Figure 6.19 Accuracy of the system for recommending contents

Figure 6.20 shows the results of accuracy vs time. The figure shows the accuracy of recommendation of devices and time taken by the switching of devices. The accuracy of device recommendation is high as shown in figure due to the reason that the location aware services perform best for the device recommendation.

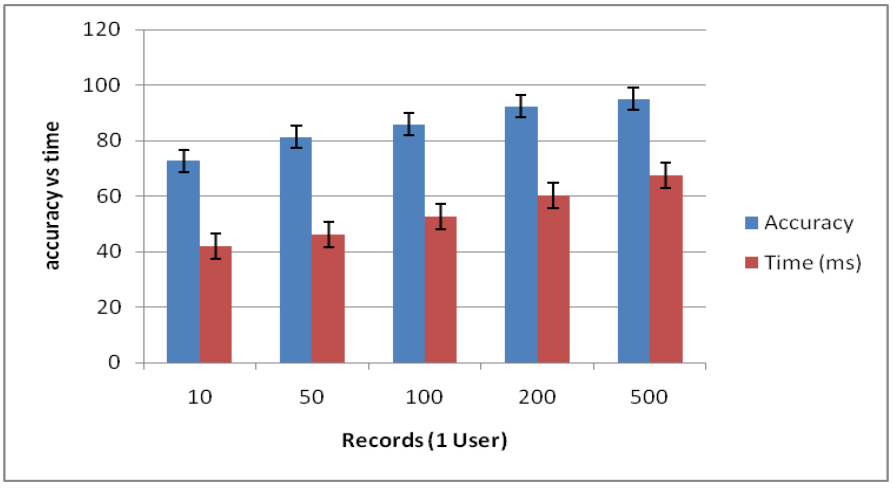


Figure 6.20 Performance comparison

Figure 6.21 shows the results of precision and accuracy graph for the recorded datasets. The results show the prominent precision and recall. However the recall is higher than the precision which means the true negative value is higher than the true positive.

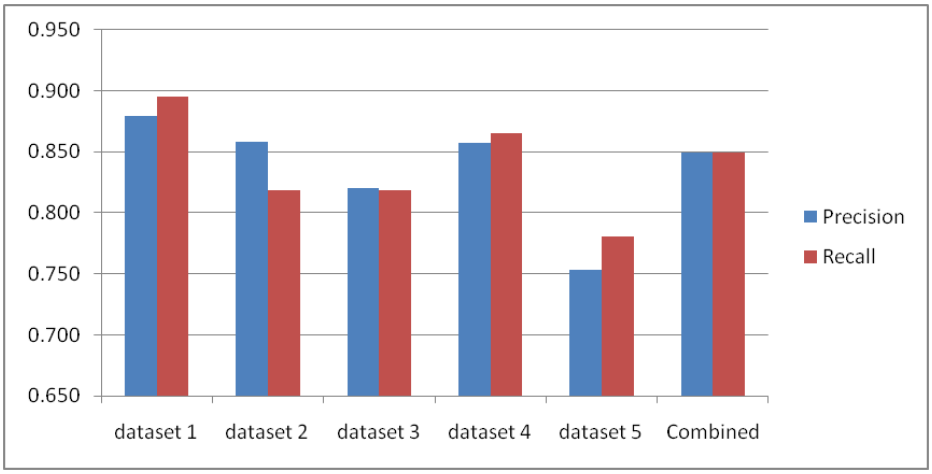


Figure 6.21: Precision vs Recall on datasets

The Figure 6.22 shows the results of all the datasets including the combination of all the datasets. Due to the reason that the movies sets are randomly recorded, therefore the TP & TN rate is almost equal which means the records are uniformly distributed.

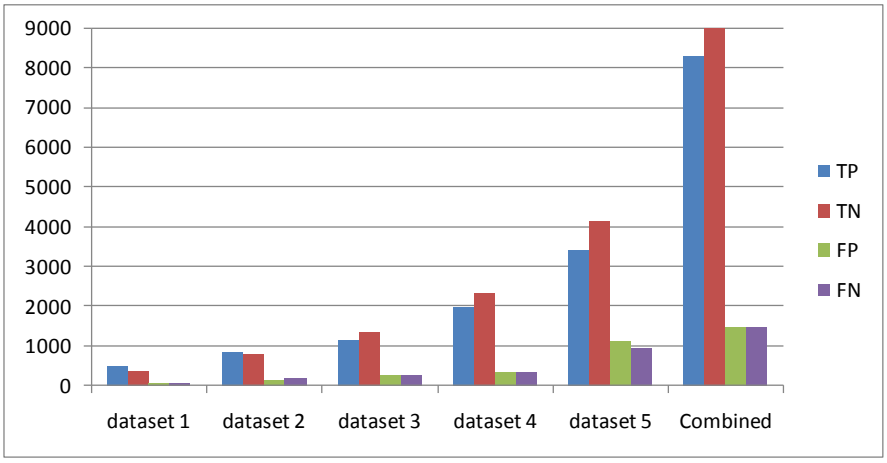


Figure 6.22: Performance comparison using TP, TN, FP, FN

Figure 6.23 shows the accuracy graph for the recorded datasets. Accuracy is almost 87% which is quite promising. The higher accuracy means the system recommend the correct content to the user.

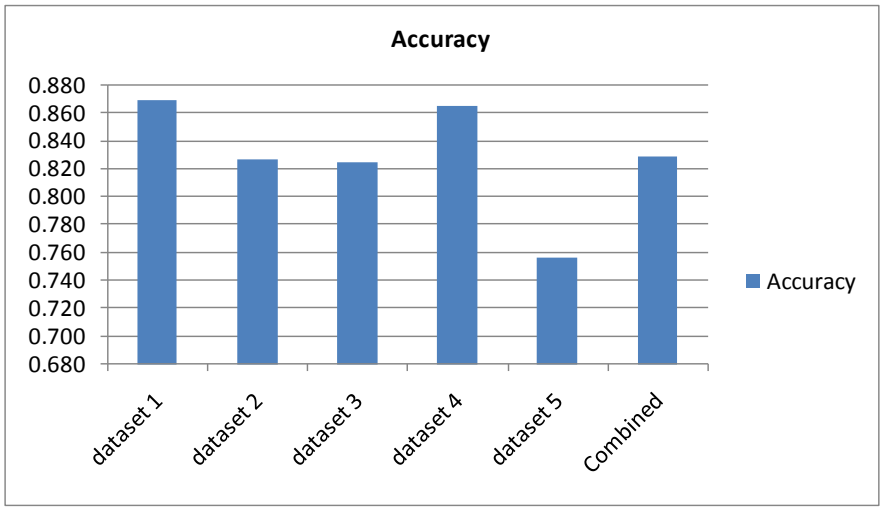


Figure 6.23: Accuracy on the different datasets

The communication delay is an important parameter in multimedia services. Therefore we calculate the response time of the user. The response time for the local app server is less than that of the overall time. This is obvious because when the contents are unavailable at the local app server then it should be downloaded. We exclude the download time and just shown a response time in both scenarios. The response time of equation1 is for the first scenario in which the content is available at the local app server.

$$R_t = \frac{1}{n}\sum_{i=0}^{n} t_n$$

Where t1 is the time between the user and local app server, t2 is the time between local app server and device gateway, t3 is the time between device and device gateway.

The following equation calculates the response time for the second scenario in which the data is not available at the local app server.

$$R_t = t_1 + t_2 + t_3 + t_4$$

Where t1 is the time between the user and local app server, t2 is the time between local app server and device gateway, t3 is the time between device and device gateway, and t4 is the time between the local app server and global content server.

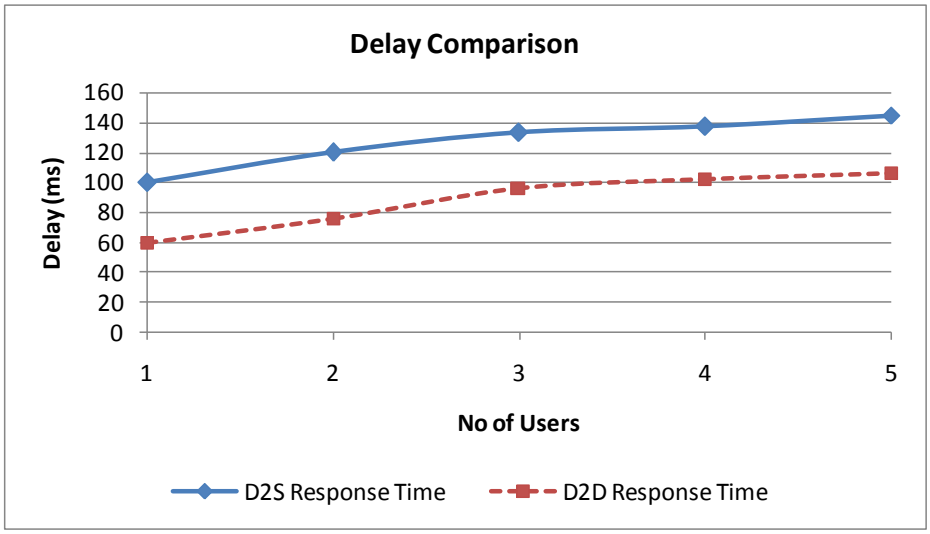


Figure 6.24: Delay comparison of devices communication

Figure 6.24 shows the response time of the system for the contents which are available on the local app server. The response time is calculated using equation mentioned above. The figure shows that the response time increases with the number of clients as the system has to entertain many requests at the same time.

Figure 6.25 shows the response time of the system on average run for 5 times. This time the system is tested with mixed option, some-time the contents are available at the local app server and some time it is missing. However the response time goes a bit higher than the first scenario. The response time is such a low that it can be ignored and hardly been observed.

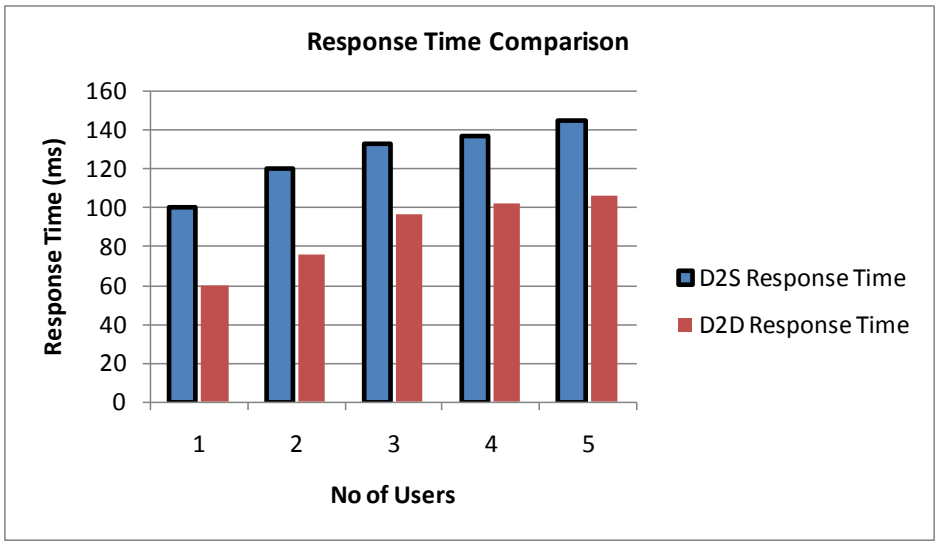


Figure 6.25: Device to Server (D2S) time comparison

6.5.2. Deep learning based recommendation results

The Netflix dataset is used which consists of two parts. The first part, with ratings and movie titles, is from the Netflix challenge dataset. The second part, with plots of the corresponding movies, was collected by [78] from IMDB 4. Similar to [94], we extract only positive ratings (rating 5) for training and testing. After removing users with less than 3 positive ratings and movies without plots, we have 407261 users, 9228 movies, and 15348808 ratings in the final dataset.

Like most recommender systems, we sort the predicted ratings of the candidate items and recommend the top M items to the target user. The recall for each user is then defined as:

$$recall = \frac{number\ of\ items\ that\ the\ user\ like\ among\ the\ top\ X}{total\ number\ of\ items\ that\ the\ user\ like}$$

The final result reported is the average recall over all users. The parameter setting is shown in Table 6.2.

Table 6.2: Parameter setting for experimentation

S.No	Parameter	Value
1	Layers	Varies (input =8000, hidden=200, output=50)
2	Lv	10
3	Lu	1
4	Ln	1e3
5	No of epoch	Varies (100~500)
6	Sparsity_cost	0.1
7	Noisedrop	0.3
8	NoOfIteration	Varies (100~500)

Table 6.3 shows the comparative results of DLM on the Netflix data set with the a 2 layer model. The algorithm shows reasonably good results on the dense dataset, however, for the sparse dataset the results are not very convincing.

Table 6.3: Recall in the sparse setting for top 300 recommendation by 2 layer model

S.No	λ_n (No of top recommended items)	Sparse recall	Dense recall
1	50	0.1	0.36
2	100	0.15	0.5
3	150	0.2	0.56
4	200	0.23	0.61
5	250	0.27	0.65
6	300	0.35	0.74

Table 6.4: Impact of layer size on the results

No of layers	1	2	3
NetflixSparse	29.20	30.50	**31.01**
NetflixDense	69.26	70.40	**70.42**

Table 6.4 shows that when the number of layers is increased to 3,the performance also increases, however the computational time is also increased with the increase in number of

layers. These primary results shows that the DLM can perform very well in the recommendation environments, however, more extensive experiments needs to be carried out with the performance comparison with the state of the art recommender techniques.

We have demonstrated in this work that state-of-the-art performance can be achieved by jointly performing deep representation learning for the content information and collaborative filtering for the ratings (feedback) matrix. As far as we know, BDLM is the first hierarchical Bayesian model to bridge the gap between state-of-the-art deep learning models and RS. In terms of learning, besides the algorithm for attaining the MAP estimates, we also derive a sampling-based algorithm for the Bayesian treatment of BDLM as a Bayesian generalized version of back propagation.

Table 6.5: Example user with recommended movies

User X	Movies in the training set: Moonstruck, True Romance, Johnny English, American Beauty, The Princess Bride, Top Gun, Double Platinum, Rising Sun, Dead Poets Society, Waiting for Gu man		
# training samples	2	4	10
Top 5 recommended movies by CTR	Swordfsh	Pulp Fiction	Best in Snow
	A Fish Called Wanda	A Clockwork Orange	Chocolat
	Terminator 2	Being John Malkovich	Good Will Hunting
	A Clockwork Orange	Raising Arizona	Monty Python and the Holy Grail
	Sling Blade	Sling Blade	Being John Malkovich
Top 5 recommended movies by BMDL	Snatch	Pulp Fiction	Good Will Hunting
	The Big Lebowski	Snatch	Best in Show
	Pulp Fiction	The Usual Suspect	The Big Lebowski
	Kill Bill	Kill Bill	A Few Good Men
	Raising Arizona	Momento	Monty Python and the Holy Grail

7. Conclusion

Due to the diverse nature of the Internet of Things, many research studies have been dedicated to address the communication architecture and data processing problem for this new paradigm. This study thus, dedicated to the enhancement in the data processing capabilities of the IoT environment with the context prediction and collaboration. The CIoT model is proposed for this purpose, which is not only capable of enhanced data processing through state of the art machine learning techniques but also addresses the issue of agreement based service collaboration.

Performance of proposed architecture is evaluated by integrating applications from three different domains. A sophisticated collaborative context prediction approach based on inductive learning is proposed and investigated. Experiments are conducted to demonstrate the effectiveness of proposed techniques on real world data sets and virtual data sets. The results show that proposed approach can be applied effectively to diverse application areas.

second application is forecasting electricity consumption in multi-family residential buildings. Current state of the art approaches for energy management in residential buildings use traditional forecasting techniques based on statistical analysis and machine learning approaches applied on monthly utility meters data, thus restricting hourly forecast. The smart metering infrastructure enables collection of the electricity consumption data at a fine temporal level. Therefore, we applied well recognized machine learning techniques i.e. neural network, regression tree and genetic programming on smart meters data to forecast electricity consumption in residential buildings.

The third application area is multimedia content recommendation in smart home environment. The home network based model is proposed for this application which proactively

responds to the user behavior in a smart home. A state of the art learning approach i.e. deep learning is used to enhance the performance of content based collaborative recommendation system. For this purpose, a Bayesian model based deep learning algorithm is used, which couples a Bayesian formulation of the stacked denoising autoencoder and probabilistic matrix factorization. Experiments on real-world datasets shows that DLM can improve the performance of the content recommendation system for sparse datasets.

References

[1] A. De Saint-exupery, “Internet of Things: Strategic Research Roadmap,” *Internet Things Strateg. Res. Roadmap*, pp. 1–50, 2009.

[2] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, “Context aware computing for the internet of things: A survey,” *IEEE Commun. Surv. Tutorials*, vol. 16, no. 1, pp. 414–454, 2014.

[3] J. He, Y. Zhang, G. Huang, and J. Cao, “A smart web service based on the context of things,” *ACM Trans. Internet Technol.*, vol. 11, no. 3, pp. 1–23, 2012.

[4] Y. Qin, Q. Z. Sheng, N. J. G. Falkner, S. Dustdar, H. Wang, and A. V. Vasilakos, “When Things Matter: A Data-Centric View of the Internet of Things,” *arXiv1407.2704 [cs]*, vol. V, no. July, pp. 1–35, 2014.

[5] F. Khodadadi, R. N. Calheiros, and R. Buyya, “A data-centric framework for development and deployment of Internet of Things applications in clouds,” *2015 IEEE Tenth Int. Conf. Intell. Sensors, Sens. Networks Inf. Process.*, pp. 1–6, 2015.

[6] A. A. Adamopoulou, A. M. Tryferidis, and D. K. Tzovaras, “A context-aware method for building occupancy prediction,” *Energy Build.*, 2015.

[7] R. Mayrhofer, “An Architecture for Context Prediction,” *Adv. Pervasive Comput.*, vol. 176, pp. 65–72, 2004.

[8] C. Voigtmann, S. L. Lau, and K. David, “An approach to Collaborative Context Prediction,” in *2011 IEEE International Conference on Pervasive Computing and Communications Workshops, PERCOM Workshops 2011*, 2011, pp. 438–443.

[9] S. Sigg, S. Haseloff, and K. David, “An alignment approach for context prediction tasks in UbiComp environments,” *IEEE Pervasive Comput.*, vol. 9, no. 4, pp. 90–97, 2010.

[10] S. Sigg, S. Haseloff, and K. David, “Prediction of context time series,” *JUUTILAINEN, M.(Hrsg.)*, pp. 31–45, 2007.

[11] “IoT Architecture - Internet of Things (IoT) Architecture.” [Online]. Available: https://standards.ieee.org/develop/wg/IoT_Architecture.html.

[12] D. R. Roberto Minerva, Abyi Biru, “Towards a definition of the Internet of Things (IoT),” *Telecom Italia S.p.A.* [Online]. Available: http://iot.ieee.org/images/files/pdf/IEEE_IoT_Towards_Definition_Internet_of_Things_Issue1_14MAY15.pdf. [Accessed: 01-Dec-2015].

[13] S. M. Alessandro Bassi, Martin Bauer, Martin Fiedler, Thorsten Kramp, Rob van Kranenburg, Sebastian Lange, *Enabling Things to Talk: Designing IoT solutions with the IoT Architectural Reference Model*, Illustrate. Springer Berlin Heidelberg, 2013, 2013.

[14] Bet. Consortium, “Building the environment for the things as a service.” [Online]. Available: http://www.betaas.eu/.

[15] “OPENIoT project description.” [Online]. Available: http://www.openiot.eu/.

[16] P. Castillejo, J.-F. Martínez, L. López, and G. Rubio, “An Internet of Things Approach for Managing Smart Services Provided by Wearable Devices,” *Int. J. Distrib. Sens. Networks*, vol. 2013, pp. 1–9, 2013.

[17] “Alexa.” [Online]. Available: URL http://www.axeda.com/.

[18] “BUGswarm.” [Online]. Available: URL http://developer.bugswarm.net/.

[19] “Carriots.” [Online]. Available: URL https://www.carriots.com/.

[20] “Etherios.” [Online]. Available: URL http://www.etherios.com/products/devicecloud/.

[21] “Evrythng.” [Online]. Available: URL http://www.evrythng.com/.

[22] “Grovestreams.” [Online]. Available: URL https://grovestreams.com/.

[23] “Nimbits.” [Online]. Available: URL http://www.nimbits.com/.

[24] “Open. Sen.se.” [Online]. Available: URL http://open.sen.se/.

[25] “Paraimpu.” [Online]. Available: URL http://paraimpu.crs4.it/.

[26] “Sensinode.” [Online]. Available: URL http://www.sensinode.com/.

[27] “SensorCloud.” [Online]. Available: URL http://www.sensorcloud.com/.

[28] “Thinkspeak.” [Online]. Available: URL https://www.thingspeak.com/.

[29] “Xively.” [Online]. Available: URL https://xively.com/.

[30] “Yaler.” [Online]. Available: URL https://yaler.net/.

[31] “Thingworx.” [Online]. Available: URL http://www.thingworx.com/.

[32] and L. T. Mohamed Ali Feki, Fahim Kawsar, Mathieu Boussard, “The Internet of Things: The Next Technological Revolution,” *IEEE Comput. Soc.*, vol. 35, no. 5, pp. 24–25, 2013.

[33] J.-S. Leu, C.-F. Chen, and K.-C. Hsu, “Improving Heterogeneous SOA-based IoT Message Stability by Shortest Processing Time Scheduling,” *IEEE Trans. Serv. Comput.*, vol. 7, no. 4, pp. 1–1, 2013.

[34] R. Want, B. N. Schilit, and S. Jenson, “Enabling the Internet of Things,” *Computer (Long. Beach. Calif).*, no. 1, pp. 28–35, 2015.

[35] P. Jäppinen, R. Guarneri, and L. M. Correia, “An applications perspective into the Future Internet,” *J. Netw. Comput. Appl.*, vol. 36, pp. 249–254, 2013.

[36] M. Kranz, P. Holleis, and A. Schmidt, “Embedded Interaction: Interacting with the Internet of Things,” *Internet Computing, IEEE*, vol. 14, no. 2. pp. 46–53, 2010.

[37] P. Bellavista, G. Cardone, A. Corradi, and L. Foschini, “Convergence of MANET and WSN in IoT urban scenarios,” *IEEE Sens. J.*, vol. 13, no. 10, pp. 3558–3567, 2013.

[38] M. Liu, Z. Li, X. Guo, and E. Dutkiewicz, “Performance analysis and optimization of handoff algorithms in heterogeneous wireless networks,” *Mob. Comput. IEEE Trans.*, vol. 7, no. 7, pp. 846–857, 2008.

[39] S. Hodges, S. Taylor, N. Villar, J. Scott, D. Bial, and P. T. Fischer, “Prototyping

connected devices for the internet of things," *Computer (Long. Beach. Calif).*, vol. 46, no. 2, pp. 26–34, 2013.

[40] S. Vinoski, "Integration with Web services," *IEEE Internet Comput.*, vol. 7, no. 6, pp. 75–77, 2003.

[41] S. Li, L. Xu, X. Wang, and J. Wang, "Integration of hybrid wireless networks in cloud services oriented enterprise information systems," *Enterp. Inf. Syst.*, vol. 6, no. 2, pp. 165–187, 2012.

[42] C. W. Tsai, C. F. Lai, M. C. Chiang, and L. T. Yang, "Data mining for internet of things: A survey," *IEEE Commun. Surv. Tutorials*, vol. 16, no. 1, pp. 77–97, 2014.

[43] L. Xu, W. He, and S. Li, "Internet of Things in Industries: A Survey," *IEEE Trans. Ind. Informatics*, vol. PP, no. 99, pp. 1–11, 2014.

[44] P. Pande and A. Padwalkar, "Internet of Things –A Future of Internet: A Survey," *Int. J. Adv. Res. Comput. Sci. Manag. Stud.*, vol. 2, no. 2, pp. 354–361, 2014.

[45] T. A. Butt, I. Phillips, L. Guan, and G. Oikonomou, "Adaptive and context-aware service discovery for the Internet of Things," *Proc. 13th Int. Conf. Next Gener. Wired/Wireless Adv. Netw. 6th Conf. Internet Things Smart Spaces (NEW2AN/ruSMART 2013)*, vol. 8121, pp. 36–47, 2013.

[46] J. Liu and W. Tong, "Adaptive Service Framework Based on Grey Decision-Making in the Internet of Things," *2010 Int. Conf. Comput. Intell. Softw. Eng.*, pp. 1–4, 2010.

[47] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio, "Interacting with the SOA-based internet of things: Discovery, query, selection, and on-demand provisioning of web services," *IEEE Trans. Serv. Comput.*, vol. 3, no. 3, pp. 223–235, 2010.

[48] M. Munoz Organero, C. Delgado Kloos, and P. Muñoz Merino, "Personalized service-oriented E-learning environments," *IEEE Internet Comput.*, vol. 14, no. 2, pp. 62–67, 2010.

[49] G. O. Talal Ashraf Butt, Iain Phillips, Lin Guan, "TRENDY- An Adaptive and Context-Aware Service Descovery protocol for 6LoWPANs," *Wot*, 2012.

[50] A. R. Biswas and R. Giaffreda, "IoT and Cloud Convergence: Opportunities and Challenges," *2014 IEEE World Forum Internet Things*, pp. 375–376, 2014.

[51] N. Bui, "Internet of Things Architecture Project Deliverable D1 . 1 - SOTA report on existing integration frameworks / architectures for WSN , RFID and other emerging IoT related Technologies," *Architecture*, no. 257521, 2013.

[52] W. Yu, G. Chen, Z. Wang, and W. Yang, "Distributed consensus filtering in sensor networks," *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 39, no. 6, pp. 1568–1577, 2009.

[53] L. Chen and J. Frolik, "Active consensus over sensor networks via selective communication," in *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2012 9th Annual IEEE Communications Society Conference on*, 2012, pp. 389–397.

[54] H. Duc Chinh, P. Yadav, R. Kumar, and S. K. Panda, "Real-Time Implementation of a Harmony Search Algorithm-Based Clustering Protocol for Energy-Efficient Wireless

Sensor Networks," *Ind. Informatics, IEEE Trans.*, vol. 10, no. 1, pp. 774–783, 2014.

[55] S. Li, L. Da Xu, and X. Wang, "A continuous biomedical signal acquisition system based on compressed sensing in body sensor networks," *IEEE Trans. Ind. Informatics*, vol. 9, no. 3, pp. 1764–1771, 2013.

[56] C.-L. Fok, C. Julien, G.-C. Roman, and C. Lu, "Challenges of satisfying multiple stakeholders," in *Proceeding of the 2nd workshop on Software engineering for sensor network applications - SESENA '11*, 2011, p. 55.

[57] J. . Choi, S. . Li, X. . Wang, and J. . Ha, "A general distributed consensus algorithm for wireless sensor networks," in *2012 Wireless Advanced, WiAd 2012*, 2012, pp. 16–21.

[58] J. R. Quinlan and R. M. Cameron-Jones, "FOIL: A midterm report," *ECML*, p. 3, 1993.

[59] J. Yin, X. & Han, "CPAR : Classification based on Predictive Association Rules," *Proc. SIAM Int. Conf. Data Mining. San Fr. CA SIAM Press*, pp. 369–376, 2003.

[60] J. P. S. Catalão, S. J. P. S. Mariano, V. M. F. Mendes, and L. a. F. M. Ferreira, "Short-term electricity prices forecasting in a competitive market: A neural network approach," *Electr. Power Syst. Res.*, vol. 77, no. 10, pp. 1297–1304, Aug. 2007.

[61] a. S. Ahmad, M. Y. Hassan, M. P. Abdullah, H. a. Rahman, F. Hussin, H. Abdullah, and R. Saidur, "A review on applications of ANN and SVM for building electrical energy consumption forecasting," *Renew. Sustain. Energy Rev.*, vol. 33, pp. 102–109, May 2014.

[62] H. X. Zhao and F. Magoulès, "A review on the prediction of building energy consumption," *Renewable and Sustainable Energy Reviews*, vol. 16. pp. 3586–3592, 2012.

[63] D. Svozil, V. Kvasnicka, and J. Pospichal, "Introduction to multi-layer feed-forward neural networks," *Chemometrics and Intelligent Laboratory Systems*, vol. 39. pp. 43–62, 1997.

[64] Y.-Q. L. R. Li, Yongchang Liu, J. -H. Li, X. -P. Gu, D. -X. Niu, "Characteristic load forecasting based on the optimized algorithm of decision tree," *Proceedings of the Chinese Society of Electrical Engineering*, 2005. [Online]. Available: http://en.cnki.com.cn/Article_en/CJFDTOTAL-ZGDC200523006.htm. [Accessed: 29-Nov-2015].

[65] I. Azmira, A. Razak, S. Majid, and H. A. Rahman, "Short Term Load Forecasting Using Data Mining Technique," *Power Energy Conf.*, no. PECon 08, pp. 139–142, 2008.

[66] A. Zeileis, T. Hothorn, and K. Hornik, "Model-Based Recursive Partitioning," *Journal of Computational and Graphical Statistics*, vol. 17, no. 2. pp. 492–514, 2008.

[67] J. R. Koza, "Introduction to genetic programming," in *Advances in Genetic Programming*, 1994, pp. 21–42.

[68] J. Wu, Y.-B. Diao, M.-L. Li, Y.-P. Fang, and D.-C. Ma, *Semi-Supervised Learning*, vol. 1, no. 2. 2006.

[69] R. E. Edwards, J. New, and L. E. Parker, "Predicting future hourly residential electrical consumption: A machine learning case study," *Energy Build.*, vol. 49, pp. 591–603, 2012.

[70] R. K. Jain, K. M. Smith, P. J. Culligan, and J. E. Taylor, “Forecasting energy consumption of multi-family residential buildings using support vector regression: Investigating the impact of temporal and spatial monitoring granularity on performance accuracy,” *Appl. Energy*, vol. 123, pp. 168–178, Jun. 2014.

[71] D. M. B. Chong Wang, “Collaborative Topic Modeling for Recommending Scientific Articles,” in *KDD’11*, 2011, pp. 448–456.

[72] Y. Hu, C. Volinsky, and Y. Koren, “Collaborative filtering for implicit feedback datasets,” *Proc. - IEEE Int. Conf. Data Mining, ICDM*, no. July, pp. 263–272, 2008.

[73] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, “Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion,” *J. Mach. Learn. Res.*, vol. 11, no. 3, pp. 3371–3408, 2010.

[74] D. J. C. MacKay, “A Practical Bayesian Framework for Backpropagation Networks,” *Neural Comput.*, vol. 4, no. 3, pp. 448–472, 1992.

[75] M. Chen, K. Q. Weinberger, F. Sha, and L. Angeles, “Marginalized Denoising Autoencoders for Domain Adaptation,” *Proc. 29th Int. Conf. Mach. Learn.*, pp. 767–774, 2012.

[76] Y. Bengio, L. Yao, G. Alain, and P. Vincent, “Generalized denoising auto-encoders as generative models,” *Adv. Neural …*, pp. 1–9, 2013.

[77] C. M. C. C. M. Bishop, “Pattern recognition and machine learning,” *Pattern Recognit.*, vol. 4, no. 4, p. 738, 2006.

[78] D.-Y. Y. Hao Wang, Naiyan Wang, “Collaborative Deep Learning for Recommender Systems,” *ARXIV*, 2014. .

[79] A. Costa, R. S. S. Guizzardi, and G. Guizzardi, “COReS: Context-aware, Ontology-based Recommender system for Service recommendation,” *... 19th Int. Conf. …*, 2007.

[80] T. Gu, H. K. Pung, and D. Q. Zhang, “Toward an OSGi-based infrastructure for context-aware applications,” *IEEE Pervasive Comput.*, vol. 3, no. 4, pp. 66–74, 2004.

[81] J. Hong, E. H. Suh, J. Kim, and S. Kim, “Context-aware system for proactive personalized service based on context history,” *Expert Syst. Appl.*, vol. 36, no. 4, pp. 7448–7457, 2009.

[82] C. Voigtmann and K. David, “A Survey To Location-Based Context Prediction,” *... on Recent Advances in Behavior Prediction …*, 2012. [Online]. Available: http://www.ibr.cs.tu-bs.de/dus-beigl/Awarecast/awarecast2012_submission_9.pdf. [Accessed: 15-Nov-2015].

[83] J. Schafer, D. Frankowski, J. Herlocker, and S. Sen, “Collaborative Filtering Recommender Systems,” *Adapt. Web*, pp. 291–324, 2007.

[84] A. Ceglar and J. F. Roddick, “Association mining,” *ACM Comput. Surv.*, vol. 38, no. 2, p. 5–es, 2006.

[85] P. Lukowicz, G. Pirkl, D. Bannach, F. Wagner, a. Calatroni, K. Foerster, T. Holleczek, M. Rossi, D. Roggen, G. Troester, J. Doppler, C. Holzmann, a. Riener, a. Ferscha, and R. Chavarriaga, “Recording a Complex, Multi Modal Activity Data Set for Context

Recognition," *Archit. Comput. Syst.*, 2010.

[86] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Förster, G. Tröster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha, J. Doppler, C. Holzmann, M. Kurz, G. Holl, R. Chavarriaga, H. Sagha, H. Bayati, M. Creatura, and J. Del R. Millàn, "Collecting complex activity datasets in highly rich networked sensor environments," *INSS 2010 - 7th Int. Conf. Networked Sens. Syst.*, no. 00, pp. 233–240, 2010.

[87] Y. Sonn, "Energy Info. Korea," 2013.

[88] G. Bebis and M. Georgiopoulos, "Feed-forward neural networks," *Potentials, IEEE*, pp. 27–31, 1994.

[89] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, vol. 19. 1984.

[90] Z. Yu, B. C. M. Fung, F. Haghighat, H. Yoshino, and E. Morofsky, "A systematic procedure to study the influence of occupant behavior on building energy consumption," *Energy Build.*, vol. 43, pp. 1409–1417, 2011.

[91] C. L. Wu, C. F. Liao, and L. C. Fu, "Service-oriented smart-home architecture based on OSGi and mobile-agent technology," *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 37, no. 2, pp. 193–205, 2007.

[92] D. Evans, "The Internet of Everything - How More Relevant and Valuable Connections Will Change the World," *CISCO Internet Bus. Solut. Gr.*, pp. 1–9, 2012.

[93] J. F. James, *A student's guide to Fourier Transforms*. 2011.

[94] K. Zhou and H. Zha, "Learning binary codes for collaborative filtering," *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discov. data Min. - KDD '12*, p. 498, 2012.