Multidimensional Analysis of Heterogeneous Relationships in Internet of Things

The University of Adelaide

A dissertation submitted in fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

Lina Yao

Supervisors: A/Prof. Michael Sheng

Prof. Anne H. H. Ngu

2014
ORIGINALITY STATEMENT

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

I give consent to this copy of my thesis, when deposited in the University Library, being made available for loan and photocopying, subject to the provisions of the Copyright Act 1968.

I also give permission for the digital version of my thesis to be made available on the web, via the Universitys digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

Lina Yao
January 10, 2014
To my parents,

my husband and my daughter
ACKNOWLEDGMENTS

I would like to thank all the people who have given me tremendous support and help to make this thesis happen.

First of all, I am deeply grateful to my supervisor, A/Prof. Michael Sheng, for his generous time and devotion on supervision and guidance during my PhD study. He instilled a thirst for excellence in me, taught me how to do high-quality research, and helped me think independently and creatively. In many ways, he has set up a great example for me: his passion on research and life, his patience in guiding students, and his diligence. I always feel lucky to have A/Prof. Michael Sheng as my supervisor, for all the earnest encouragements, and the strongest supports in every means.

I sincerely thank Prof. Anne H.H. Ngu and Assistant Professor Byron Gao, who have given me many valuable suggestions and insightful discussions in my research. I would like to thank Prof. David Suter who has been supporting me in many ways, from scholarship applications to fellowship application. I have also benefited a lot from his course on Computer Vision, which opens a fascinating research world for me.

I would like to thank my co-authors: Quan Z. Sheng, Anne H.H. Ngu, Byron Gao, Zakaria Marmarr, Xue Li, Talal Noor, Claudia Szabo, Aviv Segev, Rui Zeng for their productive and enjoyable collaborations.

I would like to also thank my friends for providing support and friendship whenever I needed. Especially thank my two dearest friends, Bo Zhao, for his unconditional support all the time, Lijuan Wang, who has kindly given me the precious help during my hardest time.

Finally and above all, I owe my deepest gratitude to my family. I want to thank my mother Junling Liu and my father Guangquan Yao for their endless and unreserved love, who have been encouraging and supporting me all the time. Their love has
provided my inspiration and has been my driving force. Without their patience and love I would have not survived the difficult times. I want to thank my husband, Tao Wang, for his love, understanding, patience, and support at every moment. I also want to thank my new born daughter Alicia, who is a miracle in my life and gives me ongoing courage to go through the challenging journey. This thesis is dedicated to them.
Abstract of the Dissertation

Multidimensional Analysis of Heterogeneous Relationships in Internet of Things

by

Lina Yao

Doctor of Philosophy in Computer Science

The University of Adelaide, 2014

The Internet of Things (IoT) describes the evolution from systems linking digital information to systems relating digital information to real world physical items. In IoT, real-world objects are being connected to the Internet and interacted with traditional web entities (images, videos, texts and people in social networks), which complicates the relationships amongst heterogeneous entities. While it is well understood that IoT offers numerous opportunities and benefits, it also presents significant technical challenges. Among them, effective and efficient management of swift growing ubiquitous things in the boosting IoT is a fundamental challenge facing society and research community today. The essential prerequisite is efficient acknowledging and analysis of heterogeneous relationships existing in Internet of Things, and with it, many research challenges can be pushed forward.

In this thesis, we have systematically explored the principles and developed a series of methodologies of mining and modeling three types of heterogeneous relationships existing in Internet of Things.

Implicit mutual correlations of things. We study the problem of discovering implicit correlations among ubiquitous things, which have unique and challenging characteristics (e.g., no uniform features, diverse, and dynamic). We explore explicit user
interactions with things and develop two graph-based models, namely T-DisCor and T-DisCor+, for discovering latent correlations among things over graphs induced from thing usage events via modeling the contextual information and entity relationships respectively. We demonstrate how the discovered correlations of things can contribute to solving a number of important applications on things management. In particular, we develop an effective and flexible feature-based method for annotating things based on the outcome of our proposed approach.

**Relationship between things and its attributes.** We investigate the problem of exploiting relationships between ubiquitous things and their different attributes, we propose multiple approaches:

- **Thing’s semantic labels.** We propose a semi-supervised learning framework for multi-label classification, which fully considers objects mutual affinity, semantic label correlation, coupled by thing-label assignment relationships for effective semi-supervised classification.

- **Thing’s social tags.** We explore social tags for web object classification by developing a tag-centric, unified and discriminative classification framework. We not only use social tags conveying partial and latent information about the web objects as a novel evidence to facilitate classifying objects on the web, but also exploit the relative information among tags.

- **Thing’s availability.** We propose an algorithm to estimate the availability of things so that those with higher availability can be selected and delivered in applications such as things composition.

**Interactive relationship between people and things.** Things recommendation is a crucial step for promoting and taking full advantage of Internet of Things, where it benefits the individuals, businesses and society on a daily basis in terms of two main
aspects. On the one hand, it can deliver relevant things (things have similar functionalities that users might need) to users based on users preference and interest. On the other hand, it can also serve to optimize the time and cost of using IoT in a particular situation. Physical things in reality have multiple unique attributes. We propose two novel approaches to explore things recommendation problem.

- We propose a probabilistic matrix factorization based joint model to address things recommendation problem in Internet of Things. We fuse information from users social networks and things correlation networks, by learning shared latent factors stemming from the probabilistic matrix factorization on three matrices, namely users relations, things correlations, and observable things usage interactions.

- We propose a novel approach that unifies collaborative filtering and content-based recommendations. In particular, our approach considers simultaneously both rating data (e.g., QoS) and semantic content data (e.g., functionalities) of services offered by things using a three-way aspect model. Unobservable user preferences are represented by introducing a set of latent variables, which is statistically estimated.

For each of model and approach, we have conducted extensive experiments to validate and evaluate it using either public data set or datasets generated from an environment we built.

The studies presented in this dissertation lead to a series of models and real-world applications, including things classification, things clustering and things recommendation. Some open research directions are also discussed in the dissertation.
# Table of Contents

1 Introduction .......................................................... 1
   1.1 Motivations and Overview .................................... 1
      1.1.1 Internet of Things ...................................... 3
      1.1.2 Heterogeneous Relationships in Internet of Things Network .. 6
      1.1.3 Why Mining Heterogeneous Relationships with Internet of Things 9
   1.2 Dissertation Organization ..................................... 10

I Learning Things Mutual Relationships .................................. 15

2 Correlation Discovery via Fusing Contextual Information in Things Usage Events .................................................. 16
   2.1 Motivations and Challenges ................................... 17
   2.2 Problem Formulation ........................................... 18
   2.3 T-DisCor .......................................................... 20
      2.3.1 Approach Overview .................................... 20
      2.3.2 Social Dimension Information Modeling ................. 21
      2.3.3 Spatial Dimension Information Modeling ................. 23
      2.3.4 Temporal Dimension Information Modeling ............. 25
      2.3.5 Correlation Inference Algorithm ........................ 25
   2.4 T-DisCor+ ....................................................... 28
      2.4.1 Approach Overview .................................... 28
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4.2</td>
<td>Spatio-temporal Fusion Graph</td>
<td>29</td>
</tr>
<tr>
<td>2.4.3</td>
<td>Periodical Pattern Detection</td>
<td>32</td>
</tr>
<tr>
<td>2.4.4</td>
<td>Inference Algorithm</td>
<td>33</td>
</tr>
<tr>
<td>2.5</td>
<td>Related Work</td>
<td>35</td>
</tr>
<tr>
<td>2.6</td>
<td>Conclusion</td>
<td>37</td>
</tr>
<tr>
<td>3</td>
<td>Things Annotation by Exploiting Correlations of Things</td>
<td>38</td>
</tr>
<tr>
<td>3.1</td>
<td>Algorithm Overview</td>
<td>38</td>
</tr>
<tr>
<td>3.2</td>
<td>Structural Feature Space</td>
<td>40</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Spectral Bisection Partitioning</td>
<td>40</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Modularity Optimization</td>
<td>42</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Structural Features Extraction</td>
<td>45</td>
</tr>
<tr>
<td>3.3</td>
<td>Posterior Probability Feature Space</td>
<td>46</td>
</tr>
<tr>
<td>3.4</td>
<td>Explicit Feature Space</td>
<td>47</td>
</tr>
<tr>
<td>3.5</td>
<td>Conclusion</td>
<td>48</td>
</tr>
<tr>
<td>4</td>
<td>Implementation and Experiments</td>
<td>50</td>
</tr>
<tr>
<td>4.1</td>
<td>T-Mine Platform</td>
<td>50</td>
</tr>
<tr>
<td>4.2</td>
<td>Implementation Details</td>
<td>53</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Sensors Hive</td>
<td>53</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Virtual Resource</td>
<td>54</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Event Processing</td>
<td>54</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Web API/Services</td>
<td>56</td>
</tr>
<tr>
<td>4.2.5</td>
<td>Rules Engine</td>
<td>57</td>
</tr>
</tbody>
</table>
4.3 A Demonstration .................................................. 58
4.4 Experiments for T-DisCor ........................................ 62
  4.4.1 Overall Performance ........................................ 62
  4.4.2 Parameters Tuning .......................................... 63
  4.4.3 Effects on Time and Location Smoothing .............. 66
4.5 Experiments for T-DisCor+ ..................................... 68
  4.5.1 Overall Performance ........................................ 69
  4.5.2 Impact of Integrating Spatio-Temporal Information .... 70
4.6 Conclusion .......................................................... 72

II Mining Things and Their Attribute Relationships 73

5 Things Classification Using Bi-Relational Graph of Things and Semantic Labels ........................................ 74
  5.1 Motivations and Challenges ................................... 75
  5.2 Problem Formulation ........................................... 78
  5.3 Bi-relational Graph Construction .............................. 80
    5.3.1 Affinity Subgraph ......................................... 80
    5.3.2 Categorical Correlation Graph ........................... 85
  5.4 Learning Algorithm ............................................. 86
  5.5 Algorithm Analysis ............................................ 90
  5.6 Experiments ..................................................... 92
    5.6.1 Experimental Settings .................................... 92
CONTENTS

5.6.2 Comparison Partners ........................................... 94
5.6.3 Experimental Results .......................................... 96
5.7 Related Work ....................................................... 99
5.8 Conclusion .......................................................... 101

6 A Discriminative Model for Things Classification by Learning Social Tags 103

6.1 Problem Formulation ............................................... 103
6.2 A Discriminative Model ............................................ 106
   6.2.1 Modeling Functions .......................................... 106
   6.2.2 Max-Margin Learning Algorithm ............................ 109
   6.2.3 Convex Optimization .......................................... 110
   6.2.4 Finding Optimal Latent Variable t' ......................... 111
6.3 Loss Function ...................................................... 112
6.4 Tag Tree Generation ............................................... 113
6.5 Discussion: Learning with Invisible Side Information ............ 115
6.6 Experiments .......................................................... 116
   6.6.1 Data Preparation ............................................. 116
   6.6.2 Tag Preprocessing ............................................ 117
   6.6.3 Performance Comparison .................................... 119
6.7 Related Work ........................................................ 121
6.8 Conclusion .......................................................... 122

7 Particle Filtering based Services Availability Estimation ................. 123

7.1 Motivations and Challenges ....................................... 124
7.2 Model Overview .............................................................. 125
7.3 Availability Modeling ....................................................... 126
7.4 The Particle Filter Based Approach .................................. 128
  7.4.1 Particle Filtering ...................................................... 128
  7.4.2 Particle Filtering Based Estimation .............................. 132
  7.4.3 The Dynamic Tracking Algorithm ............................... 134
7.5 Experiments .................................................................. 136
7.6 Related Work ................................................................ 140
7.7 Conclusion .................................................................... 142

III Analysis on Things Recommendation Relationship .......... 143

8 Exploring Things Recommendation in Internet of Things .... 144
  8.1 Motivations and Challenges ........................................... 145
  8.2 Problem Description ..................................................... 147
  8.3 The Model .................................................................. 148
    8.3.1 Encoding Users Friendship ....................................... 149
    8.3.2 Encoding Things Correlations .................................. 150
    8.3.3 Encoding User-Things Interactions .............................. 151
    8.3.4 Model Learning ..................................................... 153
  8.4 Experiments ................................................................ 154
    8.4.1 Dataset Description .................................................. 154
    8.4.2 Performance Metrics ................................................. 155
## 8.4.3 Performance Comparison

156

## 8.4.4 Impact of Things Correlations and Social Influence

158

## 8.5 Related Work

159

## 8.6 Conclusion

161

### 9 Unified Collaborative Service Recommendation Combined with Content-based Features

163

#### 9.1 Motivations and Challenges

164

#### 9.2 Requirements in Service Recommendation

166

##### 9.2.1 Problem Description

167

#### 9.3 The Unified Service Recommendation Model

168

##### 9.3.1 Model Description

168

##### 9.3.2 Model Learning

171

#### 9.4 Dealing with Data Sparsity

174

##### 9.4.1 Data Smoothing

175

##### 9.4.2 Implicit User-Descriptor Aspect Model

175

#### 9.5 Performance Evaluation

176

##### 9.5.1 Dataset Setup

177

##### 9.5.2 Metrics

180

##### 9.5.3 Performance Comparison

181

##### 9.5.4 Impact of Markoff Ratio and Latent Variables

185

#### 9.6 Related Work

189

##### 9.6.1 CF Methods for Service Recommendation

189
CONTENTS

9.6.2 Content-based Methods for Service Recommendation . . . . . . 191
9.6.3 Hybrid Methods for Service Recommendation . . . . . . . . 192
9.7 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 193

10 Conclusion and Future Work . . . . . . . . . . . . . . . . . . . . . . . 196
10.1 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 196
10.2 Future Directions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 198

Bibliography . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 203

A Curriculum Vitae . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 218
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>An illustrative scenario of Internet of Things: Tony’s one day</td>
<td>4</td>
</tr>
<tr>
<td>1.2</td>
<td>Three relationships and corresponding schemas studied in this thesis. The upper part shows three summarized relationships with Internet of Things: Thing-to-Thing (R1:T2T); Thing-to-Attribute (R2:T2A); People-to-Thing (R3:P2T). The lower part demonstrates three schemas of T2T, T2A and P2T studied in this thesis: (a) learning mutual relationship between things (Part I) (b) mining relationship between things and its attributes (Part II) (c) predicting dyadic relationship between people and things (Part III)</td>
<td>7</td>
</tr>
<tr>
<td>2.1</td>
<td>Illustration of problem description: explicit things usage events vs. implicit correlation things</td>
<td>19</td>
</tr>
<tr>
<td>2.2</td>
<td>Three dimensional graph extracted from the things usage events: (a) social graph $G_f$, (b) spatial graph $G_s$ and temporal graph $G_t$</td>
<td>21</td>
</tr>
<tr>
<td>2.3</td>
<td>Illustration of two graphs induced from thing usage events: (a) spatio-temporal graph $G_m$ (b) social graph $G_f$</td>
<td>29</td>
</tr>
<tr>
<td>3.1</td>
<td>Using the first two eigenvectors of the modularity matrix, vertices can be represented as points on a plane. By cutting the plane with a line passing through the origin (like the dashed line in the figure) one obtains bipartition’s of the graph with possibly high modularity values [89].</td>
<td>44</td>
</tr>
<tr>
<td>4.1</td>
<td>Some devices used in T-Mine platform</td>
<td>51</td>
</tr>
<tr>
<td>4.2</td>
<td>The architecture of T-Mine platform</td>
<td>53</td>
</tr>
<tr>
<td>4.3</td>
<td>Web-based user interface</td>
<td>59</td>
</tr>
</tbody>
</table>
4.4 Screenshot of Rule Composer: a user can conveniently drag any available thing’s widget panel (left) to the Rules Editor panel (right) and start to create a new rule or modify the old rules. ................................................................. 61

4.5 Overall performance comparison on six categories ........................................ 64

4.6 Parameters tuning on six categories ............................................................. 65

4.7 Impact of smoothed time ............................................................................. 67

4.8 Impact of clustered location ....................................................................... 68

4.9 Overall performance comparison: (a) Micro-F1, (b) Macro-F1 .................... 70

5.1 Illustration of a mixed graph. The lower component is the text-affinity subgraph, where the edges indicate text-text affinity. The upper component is the label-correlation subgraph, where the edges indicate label-label correlation. The dashed edges across the two components indicate text-label assignment relationships. For example, text 17 is labeled with Celebrities, Art and Entertainment. We aim to infer the most appropriate labels for the unlabeled texts, which are the ones with question marks in the figure, for example, text 25. ................................................................. 79

5.2 One-to-one pairwise affinity graph vs. sparse graph. We randomly selected 100 vertexes from the Yahoo! Arts dataset and applied both traditional similarity graph construction (left) and sparse graph reconstruction (right). The thickness of lines is proportional to the edge weight. ................................................................. 82

5.3 Micro-F1 and Macro-F1 comparison on SVM, GH, wvRN, wvRN+RL and RMG 97

5.4 Micro-F1 and Macro-F1 of SVM, GH, wvRn, wvRN+RL and RMG on each dataset 98

5.5 Comparing subgraph construction methods on \( \ell_1 \) sparse graph and \( k\text{-nn} \) graph ................................................................. 99

6.1 Tags as Puzzles: top-5 association tags assigned by users for one store selling Lady Gaga’s album: the Fame Monster ......................................................... 105
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2</td>
<td>Tags as Bridge: transferring category information between different domains</td>
<td>105</td>
</tr>
<tr>
<td>6.3</td>
<td>Graphical representation of the decomposition of the model</td>
<td>107</td>
</tr>
<tr>
<td>6.4</td>
<td>Our Tag-Centric Model with Different Base Feature Settings: Partial Textual Features (Title Only) vs. Full Textual Feature (Title+Description)</td>
<td>120</td>
</tr>
<tr>
<td>7.1</td>
<td>Services selection based on availability prediction</td>
<td>126</td>
</tr>
<tr>
<td>7.2</td>
<td>The main processes of the particle filter technique</td>
<td>129</td>
</tr>
<tr>
<td>7.3</td>
<td>Availability of a service: actual availability vs. estimated availability</td>
<td>137</td>
</tr>
<tr>
<td>7.4</td>
<td>Availability of a composite service</td>
<td>138</td>
</tr>
<tr>
<td>7.5</td>
<td>The impact of $\sigma$ on estimated availability error of a component service</td>
<td>139</td>
</tr>
<tr>
<td>7.6</td>
<td>The impact of $\sigma$ on the availability of a composite service</td>
<td>140</td>
</tr>
<tr>
<td>7.7</td>
<td>Time for composing a composite service</td>
<td>141</td>
</tr>
<tr>
<td>8.1</td>
<td>The graph induced from users’ social networks and things correlations. The connections consist of three types of links, including users’ friendships within users’ social network, things correlations with correlation graph of things and things usage links within user-thing interactions. Our task is to predict the dyadic relationship between users and things, in other words, recommending certain things to users.</td>
<td>148</td>
</tr>
<tr>
<td>8.2</td>
<td>Graphical representation of our approach.</td>
<td>149</td>
</tr>
<tr>
<td>8.3</td>
<td>Impact of social influence and things correlations</td>
<td>159</td>
</tr>
<tr>
<td>9.1</td>
<td>Graphical representation of our probabilistic model</td>
<td>169</td>
</tr>
<tr>
<td>9.2</td>
<td>Distribution of Web services and users (computers) of the WS-DREAM dataset: the circles denote Web services at different locations and the stars represent the users.</td>
<td>178</td>
</tr>
<tr>
<td>9.3</td>
<td>QoS data collection in the WS-DREAM dataset (a) response time and (b) throughput</td>
<td>180</td>
</tr>
</tbody>
</table>
9.4 Illustration of experiment implementation: we divided rating matrix $\mathcal{R}$ into the training matrix $\mathcal{R}_t$, in which partial rating scores are masked and the rest ratings are used to train our model, and evaluation matrix $\mathcal{R}_e$, in which the masked rating value can be predicted using our validated model. 183

9.5 Recommendation performance with different markoff ratios for top 1, 5, 10 Web services. 186

9.6 Recommendation performance with different number of latent variables $N_z$ for top 1, 5, 10 Web services. 187
## List of Tables

4.1 Data collection from T-Mine platform ........................................... 52  
4.2 Performance comparison with STI and without STI .......................... 71  
5.1 Yahoo! datasets ........................................................................... 93  
5.2 Overall performance comparison .................................................. 94  
5.3 Performance comparison on each dataset ...................................... 97  
6.1 Data statistics ............................................................................... 117  
6.2 Classification performance comparison ........................................ 119  
8.1 Dataset statistics .......................................................................... 155  
8.2 Comparison with PMF, SVD++, SoRec on all categories ..................... 157  
9.1 Dataset Statistics .......................................................................... 177  
9.2 Performance Comparison with Other Approaches .............................. 184  

xx
Chapter 1

Introduction

1.1 Motivations and Overview

The Internet of Things describes the evolution from systems connecting virtual information, to systems assimilating virtual information with real world physical objects. The Internet can be seen as a mixture of traditional networks and networked smart objects that will not only co-exist but also be intimately bound up with our human world. It will be an Internet with Things, where the content and services it facilitates will be all around us, always on, everywhere, all the time [61, 129]. Given the potentially vast amount of data stream from IoT systems, an increasingly important and demanding challenge will include applications and algorithms to fuse, interpret, augment and present information from smart objects. Understanding heterogeneous relationships in IoT is an essential prerequisite to cope with this challenge.

About two decades after Mark Weiser published his seminal article [129], we are one step closer to his vision of ubiquitous computing where computing power becomes invisibly integrated into the world around us and accessed through intelligent interfaces. The main driver lies in recent advances in identification technologies such as radio frequency identification (RFID), wireless sensors, and nanotechnology, which make processing power available in smaller and smaller packages that can interact and connect. Indeed, our world is slowly evolving into an environment where every-
day things such as buildings, sidewalks, and commodities are readable, recognizable, addressable, and even controllable through the Internet [111].

While it is well understood that such a ubiquitous environment offers the capability of integrating the information from both the physical world and the virtual one, which creates tremendous business opportunities such as efficient healthcare, e-business and improved environmental monitoring, it also presents an urgent need of solutions that can effectively facilitate various important applications such as search, recommendation, annotation, classification, and mashup of things.

Understanding and acknowledging heterogeneous relationships in IoT is an essential prerequisite to reap the aforementioned benefits and potentials. However, there exist a number of key challenges to overcome:

1. In general, the communication and changes among things in IoT are usually implicit and invisible to humans, which results in incomplete approaches of modeling connections and dependencies between things. Efficient and seamless information access, exchange and manipulation between the physical world and virtual one in IoT is one of major challenges.

2. Physical things hold more distinctive attributes comparing with traditional web resources. First of all, they are functionality-oriented, which determines its extensive social attributes and highly closeness with human behaviors. Moreover, physical things have more unique and intrinsic structures and characteristics (non-duplicability, statefulness and availability etc).

3. The Web of heterogeneous relationships and inter-dependencies between entities (people and things), data, information and knowledge are growing exponentially and overloading decision systems, leading to complexity and gross inefficiencies. Introducing intelligence to mining the information that is propagated
across heterogeneous relationships and entities in Internet of Things is another main challenge to make further step on application and research for Internet of Things.

Facing aforementioned challenges and towards the solutions, this thesis systematically proposes our analysis of applying data mining and machine learning to understand heterogeneous relationships of the Internet of Things, which allows the identification of patterns and interactions between smart things and human beings. It is more important to design and validate multiple algorithms and approaches to realize such a vision. We firstly abstract three dominating heterogeneous relationships in Internet of Things, and investigate the principles and methodologies for mining these heterogeneous relationships, by leveraging the information and semantics carried by things and their links. We propose models and algorithms that can exploit such rich information hidden in relationships to develop real-world applications in Internet of Things era.

1.1.1 Internet of Things

To illustrate the conception of Internet of Things and implication of multiple relationships, we firstly take the Tony’s One Day as an illustrative story to demonstrate the typical heterogeneous interactions in Internet of Things (see Figure 1.1).

Tony’s home is a typical smart home (① in Figure 1.1), where all of home appliance already are connected and enabled to the Internet and can be acknowledged and controlled by the Web interface. Tony goes to work around 7:00 am, before he leaves for work, he usually sets the rules for air conditioner, when temperature reaches to 30°C, air conditioner in the house will be automatically turned on to cool down his house. He also sets the cleaner and washing machine to begin to work around 10:00 am.
Figure 1.1: An illustrative scenario of Internet of Things: Tony’s one day
When he arrives near to his workplace, he needs to check the parking availability through his mobile phone. It is hard to find an available one efficiently, usually people have to drive around to find it. Thanks to the Smart Parking system (② in Figure 1.1), Tony only needs to subscribe this smart parking system, he can retrieve the latest status of parking lots in real-time, in the meanwhile Tony’s location can also be collected by the parking system through his GPS. The parking planner can guide and recommend him to the most appropriate parking space.

After settling down in his office, Tony is a hydraulician and needs to monitor and track the information everyday from the reservoir about the water and other related information. Thanks to Internet of Things, even though the reservoir is in a remote suburb, Tony can easily obtain desirable information, which is harvested by the sensors installed around the reservoir and in the water, and the sensor values are transmitted to Tony’s office. Then tony can use these data for analyzing and modeling about the research (④ in Figure 1.1). Around 11:00 am, Tony received a Tweet notifying from his washing machine and vacuum that cleaning and washing are already finished.

After work, Tony goes to the Gym for his routine exercise, he wears the monitoring devices which generate his personal health data during the working out, and the data can be sent to an online medical analyzer in real-time, so that his coach can adjust Tony’s exercise plan instantly based on outcome of analyzer. Tony also likes to share his working out record with his friends on Facebook, where he authorizes his friends to access his working out records so as to share the information amongst friends (③ in Figure 1.1). When Tony drives back home, he turns the automatic warming program to control the bath warm water fill in his bath tub, so that he can have a comfortable bath right after he gets home.
1.1.2  Heterogeneous Relationships in Internet of Things Network

From Tony’s One Day, we can identify three main relationships existing in Internet of Things.

1. People-to-Thing. This is the most fundamental and prior relationship in Internet of Things. The most basic People-to-Thing relationship is things usage events, which are generated when any things are accessed or in use by people. For example, a thing usage event is generated whenever Tony begins to heat food using microwave oven. Another high level relationship of People-to-Thing is recommendation, when Tony is trying to park his car, the most appropriate parking spot is recommended to him via specific algorithms.

2. Thing-to-Attributes. This relationship is mainly referring to things and their attributes. In Figure 1.1, when Tony is trying to find an available parking spot for this car, in this process, GPS installed in Tony’s car continuously collects Tony’s car location, which is one of attribute of his car and can be used in the parking spots detecting algorithm.

3. Thing-to-Thing. There are two different levels of relationships. The most basic one is direct and explicit connections between things. For example, Tony specifies a series of rules in his smart home. Taking the simplest example, when the thermometer detects the corresponding temperature (i.e., 30°C), it will trigger air conditioner to power on and begin to cool down the house. Another one is latent and implicit connection between things, which can not be observed directly. For example, what is the connection between Tony’s vacuum and his microwave?
In this thesis, we aim at analyzing and exploiting heterogeneous relationships in Internet of Things. We extract and abstract three relationships and focus on exploring the things and the interactions amongst things and people in the IoT environment. Figure 1.2 shows the three abstracted heterogeneous relationships.

Figure 1.2: Three relationships and corresponding schemas studied in this thesis. The upper part shows three summarized relationships with Internet of Things: Thing-to-Thing (R1:T2T); Thing-to-Attribute (R2:T2A); People-to-Thing (R3:P2T). The lower part demonstrates three schemas of T2T, T2A and P2T studied in this thesis: (a) learning mutual relationship between things (Part I) (b) mining relationship between things and its attributes (Part II) (c) predicting dyadic relationship between people and things (Part III)
Chapter 1. Introduction

1. **Relationship between Thing-to-Thing (T2T).** Things may exhibit similar properties to humans when interacting in a social environment. Smart objects that communicate in co-operation with context-aware applications, may affect changes in their real-world environment which may have a significant impact on underlying networking structures. We argue there are two kinds of relationships: one is explicit connection which can be direct and observed (i.e., thermometer and air conditioner in Tony’s home). Another one is implicit connection, recall the relationship between Tony’s vacuum and his microwave. This kind of relationship is our main target, since they can not be observed and need to be learned and derived. We will explore it in Part I.

2. **Relationship between Things-to-Attributes (T2A).** Diverse information can be associated with Internet of Things. Some attributes can be attached to the things or links. For example, location attributes, either categorical or numerical, are often associated with things. Also, temporal information is often associated with things usage events. Besides, things are also born with some attributes, such as availability. We will study this relationship in Part II.

3. **Relationship between People-to-Thing (P2T).** The basic relationship between people and things are the *things usage events*, when people use or access certain things. It is like a trace, implying important contextual information, which can be exploited for deriving more complex patterns (Part I). This thesis concentrates on higher level interactions, dyadic relationship indicating recommendation relationship, with which appropriate things could be pushed and recommended to the people. We will study how the information are propagated across heterogeneous information networks to serve things recommendation in Internet of Things in Part III.
1.1.3 Why Mining Heterogeneous Relationships with Internet of Things

The Internet of Things ushers a new era, which is meant to be a network of relationships and inter-dependencies between things, data, information and knowledge. The heterogeneous relationships across things and people carry not only richer semantic meanings but also can diffuse and propagate important information across the things and people. *Data mining* and *machine learning* approaches need to be developed to explore the rich information hidden in the heterogeneous relationships. We summarize four principles why mining heterogeneous relationships in Internet of Things is important based on our current studies:

- **Richer patterns hidden in people-things activities.** With more and more physical things are joining the Internet, relationships between people and smart objects are undeniably becoming an intimidating task. Because human behavior in using things is *not completely random*, and they exhibit patterns in their activities. For example, different things are used by the similar people at the time may be similar. In particular, we show the power of exploiting things usage activities, which can be leveraged to things latent correlation analysis in Part I.

- **Mining tasks by exploring intrinsic features of things.** Physical things are real entities in Internet of Things, and still carry some *intrinsic* unique features different from digital resources in Internet of Things. Studying these features is a new topic for research and development. In particular, we show our study on how to handle one of this intrinsic feature, availability, in Part II.

- **Information propagate across people network and things network.** People already form complex social networks, which are maintained and evolve by agreement on common objectives or shared interests and values. Links among
people are well explored, however, the new thought is how to propagate information across people and things in a holistic merged network containing people and things simultaneously. In particular, we present our preliminary study on this interesting issue in Part III.

- **Social-aware exploration of Internet of Things.** Things in reality have more close bonds with people comparing with digital resource. It means mining relationships in Internet of Things needs to pay more attention on human sociality, since human are social beings. *People bestow richer social attributes to things.* In particular, we study how the social tags contribute to things classification mining in Part I and Part III.

## 1.2 Dissertation Organization

The first chapter introduces the overview of the motivations and mining tasks with Internet of Things. After that, this thesis is organized into three parts largely targeting three heterogeneous relationships, which present models and algorithms for fulfilling mining heterogeneous relationships with Internet of Things. Finally, Chapter 10 concludes this thesis and discusses a few open research directions in this domain. The Chapters 2-9 are summarized as follows.

In **Part I: Learning Correlational Network of Things (CNT)**, we introduce a graph-based approach for discovering things mutual correlations using graph-based modeling methodology to form Correlational Network of Things (CNT). Under this framework, the mining task such as things annotation and things clustering can be addressed by systematic exploration of CNT.

- **Chapter 2: Correlation Discovery via Fusing Contextual Information in Things Usage Events.** Correlation discovery for ubiquitous things is critical for
many important applications such as things search, recommendation, annotation, classification, clustering, and composition. We propose a model $T\text{-}\text{DisCor}$ for discovering things mutual correlation based on user, temporal, and spatial information captured from usage events of things by using social graph, temporal graph and spatial graph [138]. Later, we extend and enhance our model as $T\text{-}\text{DisCor}+$ by using a spatio-temporal graph integrate thing usage contextual information by introducing periodical patterns [140]. We apply random walks with restart on these graphs to compute the correlation among things. The computed correlations form the Correlational Network of Things (CNT).

- **Chapter 3: Supervised Things Annotation Framework.** This framework aims at evaluating $T\text{-}\text{DisCor}$ and $T\text{-}\text{DisCor}+$ models. We design an effective and flexible feature-based method by learning a one-vs-all binary SVM classifiers for annotating things based on the outcome of our proposed model in Chapter 2. To do so, a fundamental issue is to identify and extract a number of descriptive features for each thing. We extract three types of features, structural feature, label posterior probability feature and explicit feature from CNT to form a distinctive feature spaces, and then they are fed into a discriminative classifier for multi-label classification [138, 140].

- **Chapter 4: Implementation and Experiments.** We present the experimental results of things annotation algorithm proposed in Chapter 3 on our platform. Some implementation details of our platform, $T\text{-}\text{Mine}$ and its Web-based GUI $T\text{-}\text{Mine VT}$, are also described, which is a comprehensive platform and developed for Internet of Things studies.

In **Part II: Mining Thing and Its Attributes Relationships**, we explore the relationship between things and their attributes including inherent attribute (e.g., availability)
and attached attributes (e.g., semantic labels and social tags). By investigating the relationship, mining task such as things classification, social classification and dynamic tracking can be comprehensively studied.

- **Chapter 5: Things Classification Using Bi-relational Graph of Things and Semantic Labels.** Labels of things carry rich semantic meanings, and disclose important information of things. In this chapter, we propose a novel Bi-relational heterogeneous graph where both things and their corresponding labels form a integrating graph via additional bipartite graph induced from label assignments of things.

- **Chapter 6: Things Classification by Learning Social Tags.** Social tag can be treated as a kind of social information attaching on the things by people, which reveals things’ users behavior and preference and can be used in social classification. In this chapter, we explore how well the social information attached to things can contribute to the things classification problem. We propose a discriminative model by jointly modeling the relationship between things categorical information and their corresponding social tags and uncover the relevance among social tags [139].

- **Chapter 7: Particle Filtering based Services Availability Estimation.** Different from traditional Internet entities, services provided by physical things have a unique inherent attribute, which is highly dynamic. We can proactively take corresponding actions if things availability can be predicted. In this chapter, we develop a particle-based approach to dynamically predict services availability in real-time, and dynamically maintain a subset of things with higher availability ready to join any other scenario, such as composition etc [141].

In *Part III: Analysis on Things Recommendation Relationship*, large amount of
physical things are being connected to the Internet, the interactions between people and things are becoming one of main relationships in Internet of Things. We focus on exploring dyadic relationship between people and things in Internet of Things.

- **Chapter 8: Exploring Things Recommendation in Internet of Things.** In this chapter, we propose a framework showing that the information contained in user-thing dyadic interactions is highly correlated and mutually affected with user-user connections and thing-thing correlations [137, 73].

- **Chapter 9: Unified Collaborative and Content-Based Service Recommendation.** In this chapter, we propose a probabilistic generative model, which naturally unifies the ideas of collaborative filtering and content-based methods for things recommendation [142].

The three parts correspond to three kinds of relationships summarized in Section 1.1.2. It is noted that these relationships are interdependent and not isolated. For example, in Part I, we aims to derive the implicit correlations between things and harness the primary relationship between people and things embodied in things usage events. In Part II, relationship between things definitely has distinctive effect in mining relationship between things and their attributes. In Part III, the dyadic interactions between people and things being marked as recommendation, and the recommendation process is fulfilled via propagating information across two heterogeneous relationships in things correlations and people friendships.
I

Learning Things Mutual Relationships
Chapter 2

Correlation Discovery via Fusing Contextual Information in Things Usage Events

In this chapter, we study a crucial and fundamental problem on discovering implicit correlations among ubiquitous things, which possess unique and challenging characteristics (e.g., no uniform features, diverse, and dynamic) [61, 138]. We propose to investigate explicit user interactions with things and develop the graph-based approaches for discovering latent correlations among things over graphs induced from thing usage events.

In a nutshell, the main contributions of our work are three-fold:

- We study the problem of discovering implicit correlations among ubiquitous things, which have unique and challenging characteristics (e.g., no uniform features, diverse, and dynamic). We propose to investigate explicit user interactions with things and develop two graph-based approaches $T-\text{DisCor}$ and $T-\text{DisCor}+$ for discovering latent correlations among things over graphs induced from things usage events.

- We demonstrate how the discovered correlations of things can contribute to solv-
ing a number of important applications on things management in the ubiquitous environment. In particular, we develop an effective and flexible feature-based method for annotating things based on the outcome of our proposed approach.

- We establish and develop an Internet of Things platform, T-Mine, where things are tagged by RFID and sensors, and things usage events are captured and collected in real-time. Using this real-world data with ~20,000 records collected from the testing environment over four months, we conduct extensive experimental studies to demonstrate the utility of our proposed approach.

### 2.1 Motivations and Challenges

Internet of Things provides a ubiquitous environment, which offers the capability of integrating the information from both the physical world and the virtual one, which creates tremendous business opportunities such as e-business, independent living of elderly persons and improved environmental monitoring, it also presents significant challenges \[37, 61\]. With many things connected and interacted over the Internet, there is an urgent need to effectively facilitate various important applications such as search, recommendation, annotation, classification, clustering, and mashup of things, as well as to reveal interesting patterns from things.

Discovering correlation among things is a fundamental task, serving as a foundation and necessary step for such applications on ubiquitous things. However, things correlation analysis remains a significant challenge due to the unique characteristics of ubiquitous things:

- Things cannot be easily represented in a meaningful feature space. They usually only have very short textual descriptions and lack a uniform way of describing the properties and the services they offer \[61, 25\].
• Things are normally diverse and heterogeneous in terms of functionalities, access methods or descriptions. For example, some things have meaningful descriptions while many others do not [25]. As a result, it is quite challenging to discover the explicit relationships among heterogeneous things.

• Correlations among things are not obvious and difficult to discover. Unlike social networks of people, where users have observable links and connections, things often exist in isolated settings and the interconnections between them are typically limited.

### 2.2 Problem Formulation

Correlations among things are *implicit* but we argue that they can be captured by exploring regularities in user interactions with things. Our study builds on the theory of *homophily* from the field of social networks, which suggests that people with similar characteristics tend to form relationships [86]. Then, the presence of relationships among people can be used to infer their similarities. Moreover, the stronger the tie, the higher the similarity. This inference is particularly useful when characteristics of people are not directly observable or incomplete. We advocate that the homophily principle applies to things as well, that is, things with strong interactive relationships tend to have strong correlations.

Regularities in user interactions with things can be mined from *thing usage events*, which can be explicitly captured, e.g., via RFID and sensor readings. The problem targeted in this paper can be therefore formulated as discovering *latent correlations* among things by exploiting observable thing usage events. A thing usage event happens when a person interacts with a particular thing (see lines in Figure 2.1). Let \( O = \{o_1, \ldots, o_n\}, \ U = \{u_1, \ldots, u_m\}, \ Ts = \{ts_1, \ldots, ts_p\} \) and \( \text{Loc} = \{loc_1, \ldots, loc_q\} \)
Figure 2.1: Illustration of problem description: explicit things usage events vs. implicit correlation things represent the set of things, users, timestamps and locations, respectively. A usage event of a thing $o_i$, denoted by $h \in H = \{h_1, ..., h_i\} = \{< o, u, ts, loc > | o \in O \land u \in U \land ts \in Ts \land loc \in Loc\}$, indicates that user $u$ used a particular thing $o$ located in a specific location $loc$ at a particular time $ts$. Note that thing usage events carry three dimensional information: location, time, and user, which provide rich information to discover implicit correlations among things (dotted lines in Figure 2.1). We present a novel graph-based approach to derive implicit correlations among things by mining the history of observable things’ usage events.
2.3 T-DisCor

In this section, we propose a model for discovering things correlations, namely T-DisCor (pronounce /ti : diskou/), which captures the correlation of each pair of things by considering regularities of user behaviors with similar things, social influence, as well as geographical influence. Accordingly, we propose a set of graph representations containing three dimensional information that appropriates the thing-user, thing-time, and thing-location relationships from user interacting activities. The Random Walk and Restart technique [121] is then applied on these graphs to estimate the relevance between things, which is then used to construct the Correlational Network of Things (CNT).

2.3.1 Approach Overview

we give an overview of our approach in correlation discovery of things. There are two main stages, as indicated in the overall algorithm of our approach shown in Algorithm 1. The first stage centers on building three graphs from thing usage events in terms of social dimensional information, spatial dimension information and temporal dimension information. As illustrated in Figure 2.2, captures the social influence among users on interacting things, and capture the relations between things and their temporal and geographical influence.

In the second stage, our goal is to derive the pairwise relevance scores for things. To achieve this, a random walk with restart (RWR) [132] is performed on the two constructed graphs. A relevance score is produced for any given node to any other node in the graph, presented as a converged probability. The value of the relevance score reflects the correlation strength of a pairwise things. Based on the relevance scores, a top-\( k \) correlation graph of things can be constructed, upon which many advanced
things management issues such as annotation can be solved by tapping the wealth of literature in graph algorithms.

**Algorithm 1:** T-DisCor

**Input:** Things' usage events $T$, user friendship matrix $F_u$

**Output:** Correlation matrix of things $R$

1. **Stage 1: Graphs Construction**
   2. Constructing social graph $G_f$;
   3. Constructing spatial graph $G_s$;
   4. Constructing temporal graph $G_t$;

2. **Stage 2: Inferencing correlations via traversing graphs $G_f$, $G_s$ and $G_t$**
   5. Obtaining transition probability matrix $P_f$, $P_s$ and $P_t$ deduced from weight matrix $W_f$, $W_s$ and $W_t$ respectively;
   6. Performing Random Walk with Restart over $G_f$, $G_s$ and $G_t$ to derive things correlation matrix $R_f$, $R_s$ and $R_t$;
   7. Calculating weighted linear combination correlation matrix of things $R = \alpha R_f + \beta R_s + (1 - \alpha - \beta) R_t$.

![Three dimensional graph extracted from the things usage events](image)

**Figure 2.2:** Three dimensional graph extracted from the things usage events: (a) social graph $G_f$, (b) spatial graph $G_s$ and temporal graph $G_t$

### 2.3.2 Social Dimension Information Modeling

A social graph is an augmented bipartite graph representing user interactions with things based on thing usage events. Shown in Figure 2.2 (a), such a graph contains
two sets of entities, users \( U \) and things \( O \). There is one type of intra-relation between users (also called social connections) and one type of inter-relations: edges between users and things that can be obtained from usage events. Formally, the social graph is defined as the following:

**Definition 1 (Social Graph).** A social graph, denoted by \( G_f = \langle V_f, E_f \rangle \), is an augmented undirected bipartite graph. Here \( V_f = U \cup O \) where \( U, O \) are the sets of users and things respectively. Edges \( E_f = E_U \cup E_M \), where \( E_U = \{(u, u'): (u, u') \in U \times U\} \) denotes the user social links (friendship) and the weight of each edge \( E_U(i, i') \in E_U \) is associated with the similarity between user \( u_i \) and user \( u_{i'} \). \( E_M = \{(u, o): (u, o) \in U \times O\} \) and the weight of each edge between users and things \( E_M(i, j) \in E_M \) is associated with the frequency that thing \( o_j \) is accessed by user \( u_i \).

The corresponding weight matrix \( W_f \) of graph \( G_f \) can be formulated as:

\[
W_f = \begin{bmatrix} W_U & W_M \\ W_M^T & W_O \end{bmatrix} \tag{2.1}
\]

The entries in Equation 2.1 can be obtained as follows: \( W_M \) and its transpose \( W_M^T \) should be proportional to the times of a thing being used by the users. \( W_O \) should be 0 initially since we do not consider relationships between things. The weight \( W_U \) of edges \( E_M \) indicates the user similarity influenced by the social links between users, reflecting the homophily (i.e., similar users may have similar interests). We use the cosine similarity to calculated \( W_U \) as follows:

\[
W_U(i, j) = \frac{e^{\cos(b(i), b(j))}}{\sum_{k \in \Omega(i)} e^{\cos(b(i), b(k))}} \tag{2.2}
\]

where \( \cos(b(i), b(j)) = \frac{b(i) \cdot b(j)}{||b(i)|| ||b(j)||} \), \( \Omega(i) \) is the set of the user \( i \)'s friends (i.e., \( j \in \Omega(i) \)), \( b(i) \) is the binary vector of things used by user \( i \), \( || \cdot || \) is the L-2 norm of
vector $b(\cdot)$, and $\alpha$ is a parameter that reflects the preference for transitioning to a user who interacts with the same things.

### 2.3.3 Spatial Dimension Information Modeling

A spatial graph (see the second graph of Figure 2.2 (b)) reflects the spatial information of the usage of things. We argue that geographical influence to user activities cannot be ignored, i.e., a user tends to interact with things nearby rather than the ones distant [7, 144, 24]. For example, if a user is at her office, she has higher probability of using office facilities such as telephone, printer, and seminar rooms.

**Definition 2 (Spatial Graph).** A spatial graph, denoted by $G_s = <V_s, E_s>$, is an augmented undirected bipartite graph. Here $V_s = U \cup O$ where $Loc$, $O$ are the sets of locations and things respectively. Edges $E_s = E_{Loc} \cup E_Y$, where $E_{Loc} = \{(loc, loc') : (loc, loc') \in Loc \times Loc\}$ and the weight of each edge $E_{loc}(i, i') \in E_{Loc}$ is associated with the similarity between location $i$ and $i'$, $E_Y = \{(loc, o) : (loc, o) \in Loc \times O\}$ and the weight of each edge $E_Y(i, j) \in E_Y$ is associated with the usage probability that thing $o_j$ in location $loc_i$ is accessed.

The corresponding weight matrix $W_s$ of graph $G_s$ can be formulated as:

$$W_s = \begin{bmatrix} W_{Loc} & W_{Y} \\ W_{Y}^T & W_{O} \end{bmatrix}$$

**W**$\_ Loc$ indicates the similarity of each pair of locations. Considering the usage pattern of things in specific locations. Given two locations, we measure their similarity using the Jaccard coefficient between the sets of things used at each location:

$$W_{Loc}(i, j) = \frac{|\Gamma^o_i \cap \Gamma^o_j|}{|\Gamma^o_i \cup \Gamma^o_j|}$$

where $\Gamma^o_i$ and $\Gamma^o_j$ denote the set of used things at location $i$ and location $j$ respectively.
WO should be 0 initially since we do not consider the relationships between things. WY and its transpose YT are real numbers, indicating how possible a certain thing is accessed in a certain location.

Inspired by the recent work on social networks (e.g., [7]) and location-based services (e.g., [143]), we propose to use power law distribution to model a thing’s usage probability between the thing and the user, WY ∝:

\[ z = \theta \cdot x_1^b + (1 - \theta) \cdot x_2^d \]  

(2.5)

where b and d are parameters of a power law distribution, x1 and x2 denotes the distance between user ui and thing tj and the frequency of this usage, respectively. θ is the weight for distance and frequency. z is the thing’s usage probability. Equation 2.5 can be transformed in log-log scale to fit a linear model, into the following equation:

\[ \log z = \log (\theta) + b \log x_1 + \log (1 - \theta) + d \log x_2 \]  

(2.6)

Let z’=log z, x’1=log x, x’2=log x, log (θ) as w0, b as w1, log (1-θ) as w2 and d as w3, we can have:

\[ z' = w^T x \]  

(2.7)

where w0, w1, w2, and w3 are the linear coefficients, which are collectively denoted by w, and x = \{x’1, x’2\}. In order to avoid over-fitting, we approach the weight coefficients (i.e., w) by least square error method and add a penalty term to discourage the coefficients from reaching large values, as the following:

\[ \tilde{E}(w) = \frac{1}{2} \sum_{i=1}^{n} (z'_i - z''_i)^2 + \frac{\lambda}{2} ||w||^2 \]  

(2.8)

where E(w) denotes the loss function, n represents the cardinality of input dataset, z''_i is the ground truth corresponding to z'_i, and λ is the regularization term. The optimal values of power law parameters (i.e., b, d in Equation 2.5) form the setting that minimizes the loss function E(w). In our work, we use stochastic gradient descent to solve this optimization problem [14].
2.3.4 Temporal Dimension Information Modeling

Temporal information is another important contextual indicator in the things usage events. In general, the timing of access of similar things may be similar. For example, restaurants are likely visited by people during lunch or dinner times. We therefore build a thing-time graph to capture the similarity between things in the temporal dimension. A temporal graph (see the third graph of Figure 2.2 (c)) is formally defined as the following:

**Definition 3 (Temporal Graph).** A temporal graph, denoted by $G_t = <V_t, E_t>$, is a bipartite graph. Here $V_t = T_s \cup O$ where $T_s, O$ are the sets of timestamps and things respectively. Edges $E_t = E_Z$, where $E_Z = \{(ts, o) : (ts, o) \in T_s \times O\}$ and the weight of each edge $E_Z(i, j) \in E_Z$ is associated with the frequency that thing $o_j$ is accessed in time interval $ts_i$.

And its corresponding weight matrix $G_t$ of graph $G_t$ can be formulated as:

$$W_t = \begin{bmatrix} W_O & W_Z \\ W_Z^T & W_O \end{bmatrix} \quad (2.9)$$

2.3.5 Correlation Inference Algorithm

After $G_t$, $G_f$ and $G_s$ are constructed, we can perform the random walk with restart (RWR) [132] to derive the correlations between each pair of things. RWR provides a good relevance score between two nodes in a graph, and has been successfully used in many applications such as automatic image captioning, recommender systems, and link prediction. The goal of using RWR in our work is to find things with top-$k$ highest relevance scores for a given thing. The values of the relevance scores imply the strength of the correlations among things. In the following, we focus on demonstrat-
ing RWR process over the social graph $G_f$ for discovering correlations between things. We assume the random walker starts from a thing’s node $o_i$ on $G_f$. The random walker iteratively transmits to other nodes which have edges with $o_i$, with the probability that is proportional to the edge weight between them. At each step, $o_i$ also has a restarting probability $c$ to return to itself. We can obtain the steady-state probability of $o_i$ by visiting other vertex $\pi_i$ when the RWR process is converged. The RWR process can be formulated as:

$$\pi_i = (1 - c)P\pi_i + ce_i \tag{2.10}$$

where $\pi_i \in \mathbb{R}^{N \times 1}$, weight matrix from graph $G_f$ is $W_f \in \mathbb{R}^{N \times N}$, $e_i \in \mathbb{R}^{N \times 1}$ with $i$-th entry is 1, all other entries are 0. Equation 2.10 can be further formulated as:

$$\pi_i = c(I - (1 - c)P_f)^{-1}e_i = Qe_i \tag{2.11}$$

where $I$ is an identity matrix and $P_f \in \mathbb{R}^{N \times N}$ is the transition matrix, which can be obtained based on weight matrix $W_f$ of $G_f$ by row normalization:

$$P_f = W_fD_f^{-1} \tag{2.12}$$

where $D_f$ is a diagonal matrix with $D_f(i, i) = \sum_j W_f(i, j)$. The random walker on $o_i$ traverses randomly along its edges to the neighboring nodes based on the transition probability $P_f(i, j), \forall j \in N(i)$. The probability of taking a particular edge $<o_i, o_j>$ is proportional to the edge weight over all the outgoing edges from $o_i$ based on Equation 2.12.

From Equation 2.21, we can have $Q = c(I - (1 - c)P_f)^{-1} = c\sum_{t=0}^{\infty} (1 - c)t^{t}P^t$, which defines all the steady-state probabilities of random walk with restart. $P^t$ is the $t$-th order transition matrix, whose elements $p^t_{ij}$ can be interpreted as the total probability for a random walker that begins at node $o_i$ and ends up at node $o_j$ after $t$ iterations, considering all possible paths between $o_i$ and $o_j$. The steady-state probabilities for each pair of nodes can be obtained by recursively processing Random Walk and Restart until
convergence. The converged probabilities give us the long-term visiting rates from any
given node to any other node. In this way, we can take the relevance score of all pairs
of thing nodes, denoted by \( R_f(o_i, o_j) \in R_f, \forall o_i, o_j \in O \). It should be noted that the
results can be calculated more efficiently by using the Fast Random Walk with Restart
implementation [120] via low-rank approximation and graph partition.

Similarly, the transition probability matrix \( P_t \) for the temporal graph \( G_t \), \( P_s \) for
the spatial graph \( G_s \) can be obtained respectively using:

\[
P_t = W_t D_t^{-1} \quad (2.13)
\]

where \( D_t \) is a diagonal matrix with \( D_t(i, i) = \sum_j W_t(i, j) \). Accordingly, we can
obtain the relevance scores of things on this graph \( R_t(o_i, o_j) \in R_t, \forall o_i, o_j \in O \).

\[
P_s = W_s D_s^{-1} \quad (2.14)
\]

where \( D_s \) is a diagonal matrix with \( D_s(i, i) = \sum_j W_s(i, j) \). Accordingly, we can
obtain the relevance scores of things on this graph \( R_s(o_i, o_j) \in R_s, \forall o_i, o_j \in O \).

The overall relevance score (i.e., the correlation value) of any pair of things can be
calculated using

\[
R(o_i, o_j) = \gamma R_f(o_i, o_j) + \epsilon R_s(o_i, o_j) + \eta R_t(o_i, o_j) \quad (2.15)
\]

where \( \gamma \in [0, 1], \epsilon \in [0, 1] \) and \( \eta \in [0, 1] \) are regulatory factors, which affect the
weight on the social influence and the spatio-temporal influence, and \( \gamma + \epsilon + \eta = 1 \).

With obtained correlation values, we can construct a top-\( k \) correlation graph of
things by connecting each thing with the things that have top-\( k \) overall correlation
values \( R(o_i, o_j) \), which is formally defined as the following:

**Definition 4 (Correlational Network of Things (CNT)).** A Correlational Network of
Things (CNT) is denoted by \( G = (O, E) \). For each thing \( o_i \in O \), let \( O_i^k \) denote the
top-$k$ set of correlative things to $o_i$. $E = \{e(x, i)|\forall o_i \in T, o_x \in O_1^k\}$, where $e(x, i)$ is an edge from $o_x$ to $o_i$. Each edge is associated with a weight $w_{o_x, o_i}$ with the correlation value $R_{o_x, o_i}$. □

### 2.4 T-DisCor+

In this section, we explore the contextual information in the things usage events further. We propose $T$-$DisCor+$ model by deriving three separate graphs based on three kinds of contextual information in Section 2.2. In $T$-$DisCor+$, we exploit the relationship between temporal and spatial information, and merge these two types of information in the spatio-temporal graph, which captures the spatial and temporal information in thing usage events, i.e., where and when a certain thing is accessed. In constructing this graph, we integrate the spatial and temporal information to study periodical patterns between locations and timestamps for improved performance.

#### 2.4.1 Approach Overview

In this section, we also firstly give an overview of $T$-$DisCor+$ in correlation discovery of things. There are two main stages, as indicated in the overall algorithm of our approach shown in Algorithm 2. The first stage centers on building two graphs from thing usage events. As illustrated in Figure 2.3, the spatio-temporal graph in Figure 2.3 (a) captures the relations between things and their temporal and geographical influence, while the social graph in Figure 2.3 (b) captures the social influence among users on interacting things.

In the second stage, our goal is to derive the pairwise relevance scores for things. To achieve this, a random walk with restart (RWR) [132] is performed on the two constructed graphs. A relevance score is produced for any given node to any other node.
in the graph, presented as a converged probability. The value of the relevance score reflects the correlation strength of a pairwise things. Based on the relevance scores, a top-\( k \) correlation graph of things can be constructed, upon which many advanced things management issues such as annotation and clustering can be solved by tapping the wealth of literature in graph algorithms.

Figure 2.3: Illustration of two graphs induced from thing usage events: (a) spatio-temporal graph \( G_m \) (b) social graph \( G_f \)

The spatio-temporal graph captures the spatial and temporal information in thing usage events, i.e., where and when a certain thing is accessed. In constructing this graph, we integrate the spatial and temporal information to capture periodical patterns between locations and timestamps for improved performance.

2.4.2 Spatio-temporal Fusion Graph

A spatio-temporal graph such as the one shown in Figure 2.3(a) reflects the temporal pattern and spatial information hidden in the thing usage events. In our approach, the
Algorithm 2: T-DisCor+

Input: Sequences of things usage events $T$, User friendship matrix $F_u$

Output: Correlation matrix of things $R$

1. **Stage 1: Graphs Construction**
   2. for each location $loc_i \in \text{Loc}$ do
      3. Finding time periods for $l_i$ and store as $p_i$;
      4. Constructing edges between $l_i$ and $p_i$;
   end

5. Constructing spatio-temporal graph $G_m$;
6. Constructing social graph $G_f$;

8. **Stage 2: Inferencing correlations via traversing graphs $G_m$ and $G_f$**

9. Obtaining transition probability matrix $P_m$ and $P_f$ deduced from corresponding weight matrix $W_m$ and $W_f$ respectively;

10. Implementing Random Walk with Restart (RWR) over $G_m$ and $G_f$ to derive things correlation matrix $R_m$ and $R_f$ respectively;

11. Calculating weighted linear combination correlation matrix of things $R = \theta R_m + (1 - \theta) R_f$.

Spatial and temporal information of thing usage events is treated as *inseparable* since they are mutually influential on detecting the correlations among things. Unlike virtual resources such as web pages, music or images, physical things such as restaurants and cookware usually provide the more distinguished functionalities, and more close connections with people’s daily life. So, one of distinctive feature is such physical things usually have certain physical locations and certain functioning times. For example, kitchenware are more frequently used during dining times and they have more higher likelihood to stay in a kitchen or similar locations. We specially explore the integrity between spatial and temporal information in the ubiquitous things environment via finding the periodical pattern between time and locations.

A spatio-temporal graph has three sets of nodes, namely, locations, things, and timestamps. It contains one type of intra-relation (i.e., representing similarities between locations) and three types of inter-relations between locations, things, and timestamps. Formally, we define the spatio-temporal graph $G_m$ as the following:
Definition 5 (Spatio-Temporal Graph). A spatio-temporal graph is denoted by $G_m = (V_m, E_m)$. Here $V_m = \text{Loc} \cup \text{Ts} \cup \text{O}$ where Loc, Ts and O are the sets of locations, timestamps and things respectively. Edges $E_m = E_{\text{Loc}} \cup E_{\text{X}} \cup E_{\text{Y}} \cup E_{\text{Z}}$, where $E_{\text{Loc}} = \{(loc, loc') : (loc, loc') \in \text{Loc} \times \text{Loc}\}$ and the weight of each edge $E_{\text{Loc}}(i, i') \in E_{\text{Loc}}$ is associated with the similarity between location $i$ and $i'$. $E_{\text{X}} = \{(loc, ts) : (loc, ts) \in \text{Loc} \times \text{Ts}\}$ and the weight of each edge $E_{\text{X}}(i, j) \in E_{\text{X}}$ is associated with a binary value, referring to whether location $loc_i$ has periodical relationship with time interval $ts_j$. $E_{\text{Y}} = \{(loc, o) : (loc, o) \in \text{Loc} \times \text{O}\}$ and the weight of each edge $E_{\text{Y}}(i, j) \in E_{\text{Y}}$ is associated with the frequency that thing $o_j$ in location $loc_i$ is accessed. $E_{\text{Z}} = \{(ts, o) : (ts, o) \in \text{Ts} \times \text{O}\}$ and the weight of each edge $E_{\text{Z}}(i, j) \in E_{\text{Z}}$ is associated with the frequency that thing $o_j$ is accessed in time interval $ts_i$. 

The corresponding weight matrix $W_m$ of graph $G_m$ can be formulated as:

$$W_m = \begin{bmatrix} W_{\text{Loc}} & W_{\text{X}} & W_{\text{Y}} \\ W_{\text{X}}^T & W_{\text{Ts}} & W_{\text{Z}} \\ W_{\text{Y}}^T & W_{\text{Z}}^T & W_{\text{O}} \end{bmatrix}$$ \hspace{1cm} (2.16)$$

where each entry in Equation 2.16 can be obtained as the following: $W_{\text{Loc}}$ indicates the similarity of each pair of locations. Given two locations, we measure their similarity using the Jaccard coefficient between the sets of things used at each location:

$$W_{\text{Loc}}(i, j) = \frac{|\Gamma_i^o \cap \Gamma_j^o|}{|\Gamma_i^o \cup \Gamma_j^o|}$$ \hspace{1cm} (2.17)$$

where $\Gamma_i^o$ and $\Gamma_j^o$ denote the set of used things at location $i$ and location $j$ respectively. $W_{\text{Ts}}$ and $W_{\text{O}}$ should be 0 since we do not consider the relationships between timestamps and the ones between things. $W_{\text{Y}}$ and its transpose $W_{\text{Y}}^T$ are integers, indicating how often a thing is accessed in a location. Similarly, $W_{\text{Z}}$ and its transpose $W_{\text{Z}}^T$ are integers, which indicate how often a thing is accessed at a timestamp.
For defining relationship between time stamps and locations and their corresponding weight $W_X$ of graph $G_m$, we propose periodical patterns between locations and timestamps. A periodic pattern represents the repeat of certain usage event at a specific location with certain time interval(s). Indeed, people’s interactions with things carry certain patterns. For example, people usually use a microwave oven located in a staff room between 12:00-13:00. Then a periodic pattern can be built between the location of the microwave oven and the time period of 12:00-13:00.

### 2.4.3 Periodical Pattern Detection

In this section, we will introduce how to obtain $W_X$ via detecting the periodic relationship between time and location information. Periodic patterns can be extracted by analyzing thing usage events. In our approach, we build a time series for each location where the elements of the time series are the number of time slots (e.g., 0 for the period of 0:00-1:00am; 1 for 1:00am-2:00am and so on) that a thing at a location is invoked. Given a sequence of locations, we adopt the Discrete Fourier Transform (DFT) method to detect the time periods in this discrete time-series sequence [127]. For each location, we define an integer sequence $S = \{s_1 s_2 ... s_n\}$, where $s_i = 1$ if the thing is used at this location at time $t_{s_i}$, and 0 otherwise. Essentially, this sequence can be transformed into a sequence of $n$ complex numbers $X(f)$ from the time domain to the frequency domain:

$$X(f_{k/N}) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} s(n) e^{-j \frac{2\pi kn}{N}}, \quad k = 0, ..., N - 1 \quad (2.18)$$

where $k/N$ is the frequency that each coefficient captures. As a result, DFT transforms the original sequences as a linear combination of the complex sinusoids $s_f(n) = e^{j2\pi fn/N} \sqrt{N}$. Therefore, the Fourier coefficients represent the amplitude of each of these sinusoids after sequences $S$ is projected on them.
In order to accurately capture the general shape of a time-series using a spartan representation, one could reconstruct the signal using just its dominant frequencies (i.e., the ones that carry most of the signal energy). A popular way to identify the power content of each frequency is by calculating the power spectral density (PSD) of the sequence which indicates the signal power at each frequency in the spectrum. A well known estimator of PSD is the periodogram, a vector comprised of the squared magnitude of the Fourier coefficients $X(f_{k/N})$:

$$P(f_{k/N}) = \|X(f_{k/N})\|^2, k = 0, 1, ..., \left\lfloor \frac{N-1}{2} \right\rfloor$$ \hspace{1cm} (2.19)

The $k$ dominant frequencies appear as peaks in the periodogram (and correspond to the coefficients with the highest magnitude). In order to specify which frequencies are important, we set a threshold and identify those higher frequencies than this threshold. Each element of the periodogram provides the power at frequency $k/N$ or, equivalently, at period $N/k$. That is, coefficient $X(f_{k/N})$ corresponds to periods \( \left[ \frac{N}{k}, ..., \frac{N}{k-1} \right] \). For more details, interested readers are referred to [127]. For each location, we can get its periodgram and decide the corresponding peak points based on the setting threshold. From the periodgram, we can then find the location and its corresponding time range. One benefit of using the periodgram is that we can visually identify the peaks as the $k$ most dominant periods (period $=1$/frequency). For automatically returning the important periods for a set of location sequences, we can simply set a threshold in the power spectrum to distinguish the dominant periods.

### 2.4.4 Inference Algorithm

After $G_m$ and $G_f$ are constructed, we can perform the random walk with restart (RWR) [132] to derive the correlations between each pair of things like in T-DisCor model. RWR provides a good relevance score between two nodes in a graph, and has been successfully used in many applications such as automatic image captioning, rec-
ommender systems, and link prediction. The goal of using RWR in our work is to find things with top-$k$ highest relevance scores for a given thing. The values of the relevance scores imply the strength of the correlations among things. In the following, we focus on using RWR over the spatio-temporal graph $G_m$ for discovering correlations between things. We assume the random walker starts from a thing’s node $o_i$ on $G_m$. The random walker iteratively transmits to other nodes which have edges with $o_i$, with the probability that is proportional to the edge weight between them. At each step, $o_i$ also has a restarting probability $c$ to return to itself. We can obtain the steady-state probability of $o_i$ by visiting other vertex $\pi_i$ when the RWR process is converged. The RWR process can be formulated as:

$$\pi_i = (1 - c)P_i + ce_i$$  \hspace{1cm} (2.20)$$

where $\pi_i \in \mathbb{R}^{N \times 1}$, weight matrix from graph $G_m$ is $W_m \in \mathbb{R}^{N \times N}$ (Section 2.4.2), $e_i \in \mathbb{R}^{N \times 1}$ with $i$-th entry is 1, all other entries are 0. Equation 2.20 can be further formulated as:

$$\pi_i = c(I - (1 - c)P_m)^{-1}e_i = Qe_i$$  \hspace{1cm} (2.21)$$

where $I$ is an identity matrix and $P_m \in \mathbb{R}^{N \times N}$ is the transition matrix, which can be obtained based on weight matrix $W_m$ of $G_m$ by row normalization:

$$P_m = W_mD_m^{-1}$$  \hspace{1cm} (2.22)$$

where $D_m$ is a diagonal matrix with $D_m(i,i) = \sum_j W_m(i,j)$. The random walker on $o_i$ traverses randomly along its edges to the neighboring nodes based on the transition probability $P_m(i,j), \forall j \in N(i)$. The probability of taking a particular edge $<o_i,o_j>$ is proportional to the edge weight over all the outgoing edges from $o_i$ based on Equation 2.22.

From Equation 2.21, we can have $Q = c(I - (1 - c)P_m)^{-1} = c \sum_{t=0}^{\infty} (1 - c)^t P^t$, which defines all the steady-state probabilities of random walk with restart. $P^t$ is the $t$-
th order transition matrix, whose elements $p^t_{ij}$ can be interpreted as the total probability for a random walker that begins at node $o_i$ and ends up at node $o_j$ after $t$ iterations, considering all possible paths between $o_i$ and $o_j$. The steady-state probabilities for each pair of nodes can be obtained by recursively processing Random Walk and Restart until convergence. The converged probabilities give us the long-term visiting rates from any given node to any other node. In this way, we can take the relevance score of all pairs of thing nodes, denoted by $R_m(o_i, o_j) \in \mathbb{R}_m, \forall o_i, o_j \in O$. It should be noted that the results can be calculated more efficiently by using the Fast Random Walk with Restart implementation [120] via low-rank approximation and graph partition.

Similarly, the transition probability matrix $P_f$ for the social graph $G_f$ can be obtained using:

$$P_f = W_f D_f^{-1}$$ (2.23)

where $D_f$ is a diagonal matrix with $D_f(i, i) = \sum_j W_f(i, j)$. Accordingly, we can obtain the relevance scores of things on this graph $R_f(o_i, o_j) \in \mathbb{R}_f, \forall o_i, o_j \in O$.

The overall relevance score (i.e., the correlation value) of any pair of things can be calculated using

$$R(o_i, o_j) = \theta R_m(o_i, o_j) + (1 - \theta) R_f(o_i, o_j)$$ (2.24)

where $\theta \in [0, 1]$ is a regulatory factor, which affects the weight on the social influence and the spatio-temporal influence. The Psudo algorithm for $T - DisCor^+$ is summarized in Algorithm 2.

2.5 Related Work

In this section, we review some existing works that are most closely related to our work. For mining the relevance of things and building a relational network of things,
our data source is from the users’ log-in records, which is similar to many existing research [24, 143] that exploit check-in records in Location-based Social Networks (LBSN). In our work, we also adopt the Random Walk with Restart techniques that is widely used in finding the relevance on a graph by exploiting the log-in records. For instance, [132] exploits the technique in a bi-relational network.

Relational learning refers to the classification when objects or entities present multiple relations [117]. One main technique on relational learning is based on Markov assumption, where the labels of a node in a relational network are determined by the labels of nodes in its neighborhood. Collective inference [5] and semi-supervised learning on graphs [161] work on this assumption, which is constructed based on the relational features of labeled data, followed by an iterative process (e.g., relaxation labeling method) to determine the class labels for the unlabeled data. In [5], Angelova and Weikum use it to classify the text. In [143], Ye et al. apply this methodology in Location-based Social Network for deriving the label probability for places. The authors use the collective classification method that learns the labels from the neighborhood, which only includes the nodes which hold the top-k relevance with the prediction node.

Collective inference and semi-supervised learning on graphs are limited on capturing the local dependency of nodes in the relational network. Some improvement on semi-supervised learning algorithm focus on considering the dependency between labels [78], and many other researches work on capturing the long-distance relevance of nodes. For example, in [133], Xu et al. propose a nonparametric infinite hidden relational model to capture the autocorrelation. In [88], Neville and Jensen use clustering algorithm to find cluster membership and fix the latent group variables for inference. However, both approaches, unfortunately, are not suitable for networks where the number of things might be large due to their complexity and high computational cost for
inference. Moreover, data collected from things are expected to be noisy.

2.6 Conclusion

In this chapter, we propose two novel models that derive latent correlations among things by exploiting user, temporal, and spatial information captured from thing usage events. We study the problem of discovering implicit correlations among ubiquitous things, which have unique and challenging characteristics (e.g., no uniform features, diverse, and dynamic). We propose to investigate explicit user interactions with things and develop two graph-based approaches, namely $T$-$DisCor$ and $T$-$DisCor+$, for discovering latent correlations among things over graphs induced from thing usage events. The derived mutual correlations among things form the correlational network of things (CNT), which reflects closeness and relationships among things.

This correlation analysis underpins many important applications such as things annotation, clustering, classification and search. We have performed the systematic case studies on things annotation, where the goal is to automatically assign meaningful labels to unlabeled things. The design of annotation algorithm is introduced in Chapter 3. We have implemented the annotation algorithm on our comprehensive Internet of Things platform, namely $T$-$Mine$. We report the experimental results and implementation details of $T$-$Mine$ in Chapter 4.
Chapter 3

Things Annotation by Exploiting Correlations of Things

In this chapter, we demonstrate how the discovered correlations of things can contribute to solving a number of important applications on things management. Since the CNT $G$ is essentially a graph representing the relationships among things, with the constructed $G$, many problems centered around things management (e.g., things discovery and recommendation) can be resolved by tapping sizable literature of graph algorithms. In this chapter, we will showcase an important application upon CNT: automatic things annotation.

3.1 Algorithm Overview

Automatically predicting appropriate tags (i.e., category labels) for unlabeled things can save lots of manually labeling workload, and has important research and development significance. Although some things have being labeled with useful tags (e.g., cooking, office in Figure 2.1), which are crucial for assisting users in searching and exploring new things, as well as recommending them, some things may not have any meaningful labels at all (see unlabeled things in Figure 2.1). Furthermore, a thing might associate with multiple categories. For instance, a microwave oven can
be categorized in Cooking and also Home Appliance. We propose and design a flexible feature-based classification methodology to validate our correlation discovering model. The aim of things annotation is that given a new thing, the prediction by the classifier automatically decides whether this thing belongs to the category of the corresponding labels. The algorithm can be divided in two parts: i) extracting features from top-$k$ correlation graph $G$ and things, and ii) performing the multi-label classification of things.

There are three features to extract: $F_L$, the label probabilities for testing things, and use these label probabilities vector as a feature vector. Another features $F_S$, the relationships of testing things and different thing communities and is directly extracted from the structure of CNT, the obtained communities vector indicating the relationship between the testing object and different communities of CNT, and this communities vector is use as a feature vector. $F_C$ is extracted from the descriptions of things. After obtaining the features based on attributes of $G$ and things, we combine the features $(F_L + F_S + F_C)$ together and then feed them into a Support Vector Machine (SVM) classifier or a Logistic regression function to do the supervised learning.

The overall annotation algorithm can be summarized as follows:

1. $F_L$: extracting $F_L$ via modularity-based technique to detect belonging communities of things. We take the obtained modularity vectors as features, which indicate things relationships to corresponding communities. A larger value means a closer relationship with a community.

2. $F_S$: extracting $F_S$ via generative Bayesian rules from CNT, which indicates the label probabilities for testing things. The probability vectors pointing to each label form the posterior probability feature space.

3. $F_C$: extracting content-based features of things by analyzing their textual de-
3.2 Structural Feature Space

In practice, things usually hold multiple relations. For instance, a thing might be shared among its owner, owner’s friends, co-workers, or family members. It might also be connected to other things based on functional or non-functional attributes. Detecting such relations from $G$, which can be used as a structural feature for things annotation, is naturally related to the task of modularity-based community detection [71, 90]. Modularity is a metric proposed by Newman et al. [90] to evaluate the goodness of a partition of undirected graphs, which has been shown to be an effective quantity to measure community structure in many complex networks [38, 117].

3.2.1 Spectral Bisection Partitioning

Before we go into modularity-based methodology to extract the structural features from CNT $G$, we briefly introduce the spectral bisection partitioning techniques which are closely related to modularity optimization in implementing partitioning step. Spectral bisection partitioning on graph method is based on the properties of the spectrum of Laplacian matrix [11].

Every partition of a graph with $n$ vertices in two groups can be represented by an
index vector $S$, whose component $s_i$ is +1 if vertex $i$ is in one group and 1 if it is in the other community. The cut size $R$ of the partition of the graph in the two groups can be written as:

$$R = \frac{1}{4} S^T L S$$

(3.1)

where $L$ is the Laplacian matrix. Laplacian matrix is the popularly used matrix in graph theory and it is defined as

$$L = D - W$$

(3.2)

where $W$ is the weight matrix of CNT graph $G$, whose elements $W_{ij}$ reflects the weight of the edge between vertices $i$ and $j$, $D$ is the diagonal matrix whose element $D_{ii}$ equals the degree of vertex $i$. The normalized $L$ has two main forms:

$$L_1 = D^{-1/2} L D^{-1/2}$$

(3.3)

and

$$L_2 = D^{-1} L$$

$$= I - D^{-1} W$$

(3.4)

$$= I - T$$

the sum of the elements of each row of the Laplacian is zero. This implies that $L$ always has at least one zero eigenvalue, corresponding to the eigenvector with all equal components, such as $(1, 1, ..., 1)$. Eigenvectors corresponding to different eigenvalues are all orthogonal to each other. Interestingly, $L$ has as many zero eigenvalues as there are connected components in the graph. So, the Laplacian of a connected graph has but one zero eigenvalue, all others being positive. Eigenvectors of Laplacian matrices is used for graph bipartitioning in finding optimal modularity in our work.

Vector $S$ in Equation 8.8 can be written as:

$$S = \sum_i a_i v_i$$

(3.5)
where \( v_i \) are the eigenvectors of the Laplacian. If \( S \) is properly normalized, then

\[
R = \sum_i a_i^2 \lambda_i
\]

(3.6)

where \( \lambda_i \) is the Laplacian eigenvalue corresponding to eigenvector \( v_i \).

### 3.2.2 Modularity Optimization

Considering dividing the CNT \( G \) of \( n \) vertices and \( m \) edges into \( k \) non-overlapping communities. Modularity \( Q \) is like a statistical test that the null model is a uniform random graph model, where one vertex connects to others with uniform probability. It is a measure of how far the interaction deviates from a uniform random graph with the same degree distribution. Modularity is defined as:

\[
Q = \frac{1}{2m} \sum_{ij} \left[ A_{ij} - \frac{d_i d_j}{2m} \right] \delta(s_i, s_j)
\]

(3.7)

Where \( A_{ij} \) is the adjacent matrix on the graph \( G \), \( m \) is the number of edges of the matrix, \( d_i \) and \( d_j \) denote the degree of vertex \( i \) and out-degree of vertex \( j \), and \( \delta(s_i, s_j) \) are the Kronecker delta function that takes the value 1 if node \( i \) and \( j \) belong to the same community, 0 otherwise. A larger modularity \( Q \) indicates denser within-group interaction. The modularity-based algorithm aims to find a community structure such that \( Q \) is maximized. So the Equation 3.7 can be rewritten as:

\[
Q = \frac{1}{2m} \sum_{ij} (A_{ij} - \frac{d_i d_j}{2m}) \delta(s_i, s_j) \\
= \frac{1}{4m} \sum_{ij} (A_{ij} - \frac{d_i d_j}{2m})(s_is_j + 1) \\
= \frac{1}{4m} \sum_{ij} B_{ij} s_is_j \\
= \frac{1}{4m} S^T BS
\]

(3.8)
the last expression indicates standard matrix products. The vector $S$ can be decomposed on the basis of eigenvectors $u_i, (i = 1, ..., n)$ of modularity matrix $B$, where $S = \sum_i a_i u_i$ with $a_i = u_i^T S$, then combining with Equation 3.8 with replacing with $S$:

$$Q = \frac{1}{4m} \sum_i a_i u_i^T B \sum_j a_j u_j$$

$$= \frac{1}{4m} \sum_{i=1}^n (u_i^T S)^2 \beta_i$$

(3.9)

where $\beta_i$ is the eigenvalue of $B$ corresponding to the eigenvector $u_i$. So the modularity can be optimized on bipartition via spectral bisection by replacing the Laplacian matrix with the modularity matrix [89]. Like the Laplacian matrix, $B$ always has trivial eigenvectors $(1, 1, ..., 1)$ with eigenvalue zero, because sum of the elements of each row/column of the matrix vanishes. According to Equation 3.9, if $B$ has no positive eigenvalues, which means no community structure is detected. Otherwise, the eigenvector of $B$ with positive eigenvalue $u_i$ and then group the vertices based on the signs of components of $u_i$ as same as steps like spectral partition in Section 3.2.1. In [89], Newman proposes an efficient solution by reformulating $Q$ as:

$$Q = \frac{1}{2m} S^T B S$$

(3.10)

where $S$ is the binary matrix indicating which community each node belongs to, and it can be relaxed to be continuous, the optimal $S$ is the largest $k$ eigenvectors of $Q$. $B$ is the modularity function and its entries are as follows:

$$B_{ij} = A_{ij} - \frac{d_i d_j}{2m}$$

(3.11)

Given the largest top $k$ eigenvectors, $n k -$ dimensional vectors can be constructed, each of them corresponds to a vertex. The components of the vector of vertex $i$ are
proportional to the $k$ entries of the eigenvectors in position $i$. Then community vectors for each vertex on the graph can be defined by summing the vectors of vertices in the same community. It is possible to show that, if the vectors of two communities form an angle larger than $\pi/2$ keeping the communities separate yields larger modularity than if they are merged (Figure 3.1). In this way, in a $k$-dimensional space the modularity maximum corresponds to a partition in at most $k + 1$ clusters.

Figure 3.1: Using the first two eigenvectors of the modularity matrix, vertices can be represented as points on a plane. By cutting the plane with a line passing through the origin (like the dashed line in the figure) one obtains bipartition’s of the graph with possibly high modularity values [89].

Spectral optimization of modularity is efficient and fast, the leading eigenvectors
of $Q$ can be computed with power method by repeatedly multiplying $\mathcal{B}$ by an arbitrary vector, which is not orthogonal to $u_i$. The number of iterations to reach convergence is $\mathcal{O}(n)$. The complexity of each multiplication is $\mathcal{O}(n^2)$. The time complexity of partitioning graph is $\mathcal{O}(n^2)$. To find the modularity optimal solution needs a number of subsequent bipartitions that equals the depth $\log(n)$ of the resulting hierarchical tree. To sum up, the average of time complexity to optimize $Q$ is $\mathcal{O}(n^2 \log(n))$.

### 3.2.3 Structural Features Extraction

We briefly introduce how the modularity optimization works on a unweighted and undirected graph in finding the communities containing multi-relations in Section 3.2.2. In this section, we describe extending the modularity optimization process to adapt our scenario. Since the top-$k$ correlation graph of things $G$ is a weighted and directed graph, we need to make some modifications on $Q$. This involves two steps. In the first step, we extend $\mathcal{B}$ to directed graphs. Based on [71], we rewrite the modularity matrix $\mathcal{B}$ as the following:

$$B'_{ij} = A_{ij} - \frac{d_{in}^{i} d_{out}^{j}}{2m} \quad (3.12)$$

where $d_{in}^{i}$ and $d_{out}^{j}$ are the in-degrees and out-degrees of all the nodes on $G$. In the second step, we extend $B'$ to weighted graphs. To do so, we conduct further modifications based on Equation 3.12. According to [90], it can be rewritten as:

$$B''_{ij} = \mathcal{W}_{ij} - \frac{w_{in}^{i} w_{out}^{j}}{2m} \quad (3.13)$$

where $\mathcal{W}_{ij}$ is the sum of weights of all edges on $G$ replacing the adjacency matrix $A$, $w_{in}^{i}$ and $w_{out}^{j}$ are the sum of the weights of incoming edges adjacent to vertex $o_i$ and the outgoing edges adjacent to vertex $o_j$ on the $G$ respectively.
It should be noted that different from undirected situation, $B''$ is not symmetric. To use the spectral optimization method proposed by Newman in [89], we restore symmetry by adding $B''$ to its own transpose [71], thereby the new $Q_{\text{new}}$ is:

$$Q_{\text{new}} = \frac{1}{4m} S^T (B'' + B''^T) S$$

$$Q_{\text{new}} = \frac{1}{4m} S^T B_{\text{new}} S$$  \hspace{1cm} (3.14)$$

After a series of reformulations, We still can solve Equation 3.14 using method introduced in Section 3.2.2. We are able to calculate all the eigenvectors corresponding to the top-$k$ positive eigenvalue of $B_{\text{new}}$ via the multiplication of $B_{\text{new}}$ and a vector $x$ as:

$$B_{\text{new}} x = \mathcal{W} x - \frac{w^T x}{4m} w$$  \hspace{1cm} (3.15)$$

Then the solutions are the obtained modularity vectors for each vertex, and we take them as the latent features, which indicate things relationships to communities. A larger value means a closer relationship with a community.

### 3.3 Posterior Probability Feature Space

$F_L$ indicate label probability for each given testing object as a kind of feature. It can be retrieved straightly using generative Bayesian rules from $G$, where each testing thing $o^*$ is to be assigned one or multiple labels $l_k \in L = \{l_1, ..., l_k\}$. We simply propose to formulate our solution as posterior probability $Pr(l_k|o^*)$. Once we know these probabilities, it is straightforward to assign $o_i$ with the label having the top-$k$ largest probabilities, which can be derived using Bayesian rules straightly:

$$p(l_k|o^*) = \frac{p(o^*|l_k)p(l_k)}{\sum_{j=1}^{K} p(x|l_j)p(l_j)}$$

$$Pr(l_k|o^*) \propto Pr(o^*|l_k)Pr(l_k)$$  \hspace{1cm} (3.16)$$
where the prior distribution probability \( p(l_k) \) can be calculated from the training dataset. Let \( o^k = o^k_1, ..., o^k_M \) be the training dataset, its size is \( M_k \) things with label \( k \). Then \( p(o^*|l_k) \) can be calculated using:

\[
p(o^*|l_k) = \frac{1}{Z} \sum_{m=1}^{M_k} p(o^*|o^k_m,l_k)p(o^k_m|l_k)
\]

\( (3.17) \)

where \( Z \) is a normalizing constant and the conditional probability \( Pr(o^*|o^k_m,l_k) \) indicates the relevance score between testing thing \( o^* \) and things in the training dataset \( o^k_m \). \( Pr(o^*|o^k_m,l_k) \approx \pi_{o^*} \), which are obtained in our RWR process, gets the steady state probability between \( o^* \) and \( o^k = o^k_1, ..., o^k_M \). The distribution \( p(o^k_m|l_k) \) is set as an uniform distribution \( 1/M_k \). Then we can predict the probability \( p(o^*|l_k) \) in Equation 3.17, and assign the labels with different posterior probabilities to the testing thing.

### 3.4 Explicit Feature Space

For the content features from thing descriptions, we apply an adapted Term Frequency/Inverse Document Frequency (TF/IDF) method to extract representative keywords from RESTful descriptions of things. The set of keywords is formed the explicit features of things, \( F_C \).

The tf-idf calculation assigns each term \( x \) a weight in text \( d \). \( tf/idf(x,d) = tf(x,d) \times idf(x) \), where \( tf(x,d) \), the number of times word \( x \) occurs in the corresponding text \( d \), and \( idf \) is the inverse text frequency which is defined as : \( idf(x) = \log \frac{|N|}{df(x)} \), where \( |N| \) is the number of texts in our dataset, and \( df(x) \) is the number of texts where the word \( x \) occurs at least once. So the set of feature vectors for the \( N \) text in the dataset \( \mathcal{V} = [v_1, ..., v_N] \) where \( v_i \in \mathbb{R}^m \) is the feature vector for each text.

It should be noted that the common implementation of TF/IDF gives equal weights
to the term frequency and inverse document frequency (i.e., \( w = tf \times idf \)). We choose to give higher weight to the idf value (i.e., \( w = tf \times idf^2 \)). The reason behind this modification is to normalize the inherent bias of the TF measure in short documents. Traditional TF/IDF applications are concerned with verbose documents (e.g., books, articles, and human-readable Web pages). However, documents describing things are relatively short. Therefore, the frequency of a word within a document tends to be incidental, and the document length component of the TF generally has little or no influence.

3.5 Conclusion

In this chapter, we focus on demonstrating how the discovered correlations of things deduced from Chapter 2 can contribute to solving a number of important applications on things management in the ubiquitous environment. To validate the feasibility and benefits of our proposed approaches, in particular, we design an effective and flexible feature-based algorithm for annotating things based on the outcome of T-DisCor and T-DisCor+. We treat things annotation as a feature-based multi-label classification problem, where given a new thing, the prediction by the classifier automatically decides whether this thing belongs to the category of the corresponding labels.

To do so, we extract three types of features from Correlational Network of Things (CNT) \( G \), one kind of feature is extracted from characteristics of \( G \). We adopt and extend modularity optimization technique into weighted and directed CNT \( G \), to derive community vectors for each vertex, and which repress the which communities each vertex belong to. We then deduce the label probability vector for each vertex, which indicates the label probability of each vertex might have in the entire label space. Furthermore, ubiquitous things are exposed as RESTful Web services in our implementation, and each of them has a short web description. By analyzing the textual
description using tf-idf, it is possible to extract the most common terms that represent
the corresponding thing (i.e., content-based features). With all of three features form-
ing a distinctive feature space, a one-vs-all discriminative classifier can be built up and
accomplish the things annotation. The details of implementing this things annotation
algorithm on CNT $G$ and corresponding experimental results are elaborated in next
chapter (see Chapter 4).
Chapter 4

Implementation and Experiments

This chapter is devoted to the implementation and performance study of our proposed correlation discovery models in Chapter 2, via implementing things annotation algorithm in Chapter 3 and things clustering. We implemented these techniques inside the T-Mine Platform, which aims at providing a comprehensive platform for enabling Internet of Things.

4.1 T-Mine Platform

Internet of Things (IoT) is still relatively new and it is hard to find large-scale data for our experiments. We have developed T-Mine platform that consists of several different physical places including workspace (e.g., offices, labs, seminar rooms) and one author’s home (e.g., kitchen, bedroom, living area). The physical things are enabled to Internet using RFID technology and sensors, and they are exposed on the Web via RESTful Web services, which can be discovered and accessed from a Web-based interface (e.g., Web browsers). We believe our T-Mine platform is a reasonable first step to validate the ideas proposed in Part I.

The system has two ways to identify the physical objects and connect them to the Web. One is via the RFID technology, where the physical objects are attached with RFID tags and interrogated by fixed RFID readers located in our testing environment.
The other is to combine sensors with objects to transfer the raw data to the network. The raw data captured by readers and sensors will be further processed.

For the hardware of RFID technology applied in this platform, we adopt passive RFID tag considering low cost and convenience of deployment. We also deploy various sensors for sensing motions, spatial informations, pressures and temperature etc. Figure 4.1 shows some of the RFID devices and sensors used in the implementation.

![RFID devices and sensors](image)

Figure 4.1: Some devices used in T-Mine platform

Here, we list some main kinds of sensors we use in our work as follows:

- **Motion sensors**: detecting changes in infrared radiation which occur when there is movement by a person (or object) that is different in temperature from the surroundings.

- **Spatial 3/3/3 sensors**: measuring static and dynamic acceleration in 3 axes, up to 5g, magnetic field in 3-axes up to 4 Gauss, and angular rotation in 3 axes, up to 400 per second.

- **Rotation sensors**: These sensors can be rotated 300 degrees and output a number between 0 and 1000 based on the shaft position. The maximum resistance of the potentiometer is 10K ohm.

- **GPS sensors**: The GPS sensors provides the longitude and latitude of the board’s position in signed decimal degree format. The position accuracy (best case) is 2.5m CEP (Circular Error of Probability).
Chapter 4. Implementation and Experiments

- Temperature sensors: These sensors are an intelligent non-contact temperature sensors, which output a continuous data flow every 32ms with an active alarm running in the background, and the temperature output data for distant objects ranges from -70°C to 380°C.

When users interact with things, events are captured, e.g., by RFID readers. These raw data of the events have to be preprocessed (e.g., cleaning, transformation, integration) before storing at the repository. In our experiments, we monitored 127 different physical things in six categories using RFID tags and sensors and exposed them on the Web using RESTful Web services (Table 4.1). We also manually labeled these things with 397 different labels. It should be noted that some things belong to multiple categories, therefore having multiple labels. This dataset serves as the ground-truth dataset in our experiments for performance evaluation. Ten volunteers participated in the data collection phase by interacting with RFID tagged things for a period of four months, generating 20,179 records on the interactions of the things tagged in the experiments. In the following, we briefly describe some implementation details on collecting thing usage events.

Table 4.1: Data collection from T-Mine platform

<table>
<thead>
<tr>
<th>No.</th>
<th>Category</th>
<th>Number of Things</th>
<th>Number of Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Entertainment</td>
<td>28</td>
<td>118</td>
</tr>
<tr>
<td>2</td>
<td>Office</td>
<td>20</td>
<td>51</td>
</tr>
<tr>
<td>3</td>
<td>Cooking</td>
<td>25</td>
<td>103</td>
</tr>
<tr>
<td>4</td>
<td>Transportation</td>
<td>11</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>Medicine/Medical</td>
<td>10</td>
<td>18</td>
</tr>
<tr>
<td>6</td>
<td>House Appliances</td>
<td>33</td>
<td>83</td>
</tr>
</tbody>
</table>

Figure 4.2 shows the architecture of T-Mine platform. The platform has a scalable layered architecture, and is developed in Microsoft .NET framework and SQL Server 2012 database. The physical things and their related data and events are mapped to corresponding virtual resources, which can be aggregated and visualized via a range
Chapter 4. Implementation and Experiments

4.2 Implementation Details

The T-Mine platform offers a comprehensive environment for enabling Internet of Things and supporting development of various applications on it. The main modules are described as the following.

4.2.1 Sensors Hive

This module manages sensors and RFID tags, establishes connections, and queries sensor/RFID status, thus allowing physical things to be mapped to the corresponding virtual resources. It provides a universal API so that higher level programs can retrieve status of sensors with specified address without knowing where and how to find the
physical sensor (e.g., to which port the sensor is attached). Another advantage of introducing Sensor Hive is to allow the system to provide data synchronously. This is important for some devices that work with more than one sensors since the sensor readings may come asynchronously. This module works in a scalable plug-and-play fashion, where new sensors can be plugged-in when they are ready to use and old sensors can be easily removed.

4.2.2 Virtual Resource

This module maps a collection of classes to their corresponding real devices. Each virtual device collects related sensor readings from Sensor Hive and uses the collected information to detect the current status of the corresponding real device. For example, a Microwave Oven virtual device can query its sensor values from Sensor Hive, then use these readings to decide the current status (idle or busy) of the real microwave oven attached by these sensors. The virtual devices generate events based on the readings, which can be subscribed by other programs. In our implementation, these events are subscribed by an event logger, which keeps a record in the database. Since each kind of device has a virtual resource, systems built on this architecture can be easily extended and maintained.

4.2.3 Event Processing

This module automatically extracts and aggregates things usage events based on data feeds from Virtual Resources in a pipelined fashion. The pipeline consists of three phases: event detector, contextual information retriever, and event aggregation.

Even Detector decides and captures whether a physical thing is in-use. In our implementation, there are two ways to detect usage events of things: sensor-based for detecting state changes and RFID-based for detecting mobility. In the former, the us-
age of an object is reflected by changes of the object’s status. For example, when a microwave oven is turned into working from idle, our system can determine that this oven is being used. In such a situation, we mainly adopt sensors in tracking state changes of things. In the latter, the movement of an object indicates that the object is being used. For example, if a coffee mug is moving, it is likely that the mug is being used. In this situation, we adopt a generic method based on comparing descriptive statistics of the Received Signal Strength Indication (RSSI) values in consecutive sliding windows [95]. The statistics obtained from two consecutive windows are expected to differ significantly when an object is mobile. A threshold can be set to determine whether this difference is related to a mobility and can be regarded as a valid usage event.

Contextual Information Retriever retrieves and calculates contextual information contained in things usage events. In our current implementation, we focus on three types of contextual information: identity (user), temporal (timestamp) and spatial (location) information. To obtain the identity information, we perform manual labeling and all participants need to mark and record their activities. To obtain the temporal information, we split one day into 48 equal intervals, each half an hour. For example, if the timestamp of a usage event is 9:07am, it will be assigned to the 9:00-9:30am interval. To obtain the spatial information, we consider two situations.

For static objects (e.g., refrigerator, microwave oven), the spatial information is from prior knowledge. For mobile objects (e.g., RFID-tagged coffee mug), we provide coarse-grain and fine-grain methods. The coarse-grain method uses the RSSI signal received from a tagged object to approximate its proximity to a reader antenna. Each zone is covered by a mutually exclusive set of RFID antennas. The zone scanned by the antenna with the maximum RSSI signal is regarded as the object’s location. The fine-grain method compares the signal descriptors from an object at an unknown location
to a previously constructed radio map or fingerprint. We use the Weighted $k$ Nearest Neighbors algorithm (w-kNN) to find the most similar fingerprints and compute a weighted average of their 2D positions to estimate the unknown tag location \[92\].

Event Aggregation indexes and stores all the events and services, together with their related information. It is a higher level wrapper of Database Access Object (DSO) with the purpose of decoupling business logic layer (the service layer, or other layers that need to access sensor data from the database) from underlying database operation. The indexed events and services and their corresponding contextual information are stored in a database, which can be used to contribute data mining tasks in ubiquitous environment \[138, 140\]. A list of elements are constructed, storing the identifiers of objects, their types and values, calculated contextual information, etc. In this way, applications can focus on the functionalities without worrying about operations such as connecting to the database, opening connections, querying with specified languages and handling the results (normally they are raw data and inconvenient to access). End users also can access and query data through the provided user interface.

4.2.4 Web API/Services

This module converts events and data into corresponding services. By providing RESTful APIs for things, applications can easily access data associated with a particular thing stored in the database (e.g., usage history of a device), as well as manipulate the sensors (e.g., turning on or off a light). The APIs are represented using JavaScript Object Notation (JSON) \[^1\], which developed from the JavaScript for representing simple data structures and associative objects. It is easy for machines to parse and generate, and is independent of the other programming languages.

[^1]: http://www.json.org/
4.2.5 Rules Engine

This module allows users to control the device by setting up a series of basic rules. Rule Composer contains three functionalities: a visualized rule builder, a rule interpreter and a rule parser. A rule consists of two parts: a condition part and an action part. A condition is a composition of a set of simple boolean expressions, for example, “Temperature ≥ 20 and Time=[20:00:00,23:00:00]” means when temperature (from a device) is higher than 20 and time is between 8pm and 11pm, this condition will return true. By combining simple boolean expressions together, the user can setup a complex rule to make devices “smarter”. An action is simply a set of settings of devices, for example, “OutDoorLight.On=true” will turn on “OurDoorLight” device. Alternatively, an action can also be a prepared program, for example, “SendEmail(Microwave, ‘someone@example.com’)” will send an email to “someone@example.com”. Rules Engine includes three main components: Rule Builder, Rule Interpreter and Rule Parser.

- Rule Builder is a web-based application implementing a user-friendly GUI for rule creation and action setup, where a user only needs to drag the icon of device to the editing area, then setup the details by simple clicks. After all setup is done, it will generate a string expression of the rule, then send it back to the server.

- Rule Interpreter is a program that receives the string expressions sent from the Rule Builder. It will analyze and annotate the string statement-based on a state machine. The string expression will be translated to a list of annotated objects. Each object will clearly know what type it is, such as number, build-in functions, device property, operators or bracket, etc. The outcome of the rule interpreter is a sequence of annotated inputs, which will be passed to the rule parser. The rule interpreter can also convert the string statement into a simple structure so that
the rule builder can generate a corresponding GUI via understand rules. This is important for users to modify their existing rules.

- Rule Parser is a compiler based on the *Shunting-yard algorithm*. It first compiles each part of the input sequence into a .NET Expression object. Then, it combines all such objects together into a complex Expression Tree, which will be compiled into a *Lambda Expression*. This Lambda expression object will be stored in memory when the host is running. It can be invoked when device status changes or time elapses. If the Lambda expression returns true, a corresponding action will be called.

### 4.3 A Demonstration

We have developed multiple applications and experiments on top of the T-Mine platform, including smart home, things annotation and things recommendation etc. In this section, We showcase a prototype system that offers an integrated Web-based interface to manage (i.e., connect, monitor, mashup, and visualize) things based on *T-Mine* platform, constructs a Web-based system, where the information produced by physical things can be managed and propagated through the Web and social network interfaces.

More specifically, this web-based system offers context-aware composition of things. In our work, contextual information includes environmental information (e.g., temperature, light condition), activities (e.g., a person approaching the door of a house) and social events (e.g., tweet updates from people or socialized things). Since things are socialized, they can talk, behave like humans, and respond to context naturally. We also adopt a rule-based approach to aggregate individual things and build context-aware, personalized new value-added services. We have implemented the prototype in a real-world environment, where people can access, control, and compose physical resources.
regarding the diverse contexts, just like what they do on virtual resources.

Our system demonstrates a way of integration between physical things and the virtual world in the context of ubiquitous computing and the Internet of Things. The context of a real, inhabituated home environment demonstrated in this section by both virtual resources and physical things provides the basis for various other research in specific domains, such as independent living of the elderly, healthcare and smart homes. Our system can be considered as a step further to realizing the IoT vision.

Figure 4.3 shows the Web-based user interface, including the following five main functionalities.

1. **Real-time visualizer and monitor:** This function offers access to and control of the physical things, allowing the real status of physical things to be visualized in real time (Part 1 in Figure 4.3).

2. **Twitter notification:** Twitter is utilized to notify the status of physical things to
the subscribers in real time (Part 2 in Figure 4.3). This function is very useful in practice. For example, staff members on a university campus will be able to know the status of facilities (e.g., coffee machine) via subscribing to Twitter without physically entering the facility room. In our demonstration, the status of real things (e.g., lights, ovens) in a smart home environment can be tweeted in real time to family members who subscribe to them.

3. **Query and control panel**: This function offers an interface, via which users can access more detailed and historical information about the things. An example is shown in the inset of Figure 4.3 about a light device. Users can also change the status of the things from the interface (Part 3 in Figure 4.3). For example, if a user is working overtime at the office, he can turn on the lights in his home for security reasons using his laptop or mobile phone.

4. **Rules composer**: This function provides a complex graphical interface for Rules Engine allowing users to control the devices by setting up a series of basic rules via Web browser (Part 4 in Figure 4.3). It can help create a composite service using the rule-based composition component provided by the system. It consists of **Widget Panel** and **Rules Editor Panel** as shown in Figure 4.4.

The Widget panel (left) shows the enabled objects (things and people) in the current web portal. Each thing is combined with possible actions. For example, the light is associated with two actions: turn on and turn off. As another example, the clock has a slider to set the time range.

The Rules Editor panel (right) lists all the current rules. Users just need to drag any thing’s widget to the Rules Editor panel and start to create a new rule or change old rules by clicking the **Edit** button beside each rule. An editing panel includes a condition editor (Middle) and an action editor (Right). The condition

---

[^1]: [https://twitter.com/LinaSmarthome](https://twitter.com/LinaSmarthome)
editor allows the users to change or set rules based on contextual information (e.g., environmental information such as time and temperature, event information such as people approaching). The rules will be saved to the database by clicking the “Save” button.

Figure 4.4: Screenshot of Rule Composer: a user can conveniently drag any available thing’s widget panel (left) to the Rules Editor panel (right) and start to create a new rule or modify the old rules.

5. **Object Tracing**: The learned position information from the Event Processing module is used to visualize the traces of objects in a coarse-grain level. We divide the testing area into multiple grids in a chess board style. Then, the trace line is generated by connecting the central dots of grids. For example, as shown in Figure 4.3, the trace of a coffee mug held by a subject is displayed in the blue lines. It should be noted that the visual tracking view is only activated when the mobility of a certain thing is detected.
4.4 Experiments for T-DisCor

We have conducted three experiments to validate T-DisCor approach. We focus on reporting the evaluation results for things annotations in terms of i) overall performance; ii) parameters turning on impact of $\gamma$, $\epsilon$ and $\eta$; iii) impact on time and location smoothing.

We adopt the following widely used performance metrics [153] to evaluate the performance of our proposed approach:

- **Hamming loss** evaluates how many times a thing is misclassified, i.e., a thing not belonging to a category is predicted or vice versa;

- **One-error** evaluates how many times the top ranked predicted things are not in the ground-truth dataset;

- **Coverage** evaluates how far we need to go down the list of predicted things on average in order to recover the ground-truth dataset, and

- **Average precision** evaluates the average fraction of things ranked above a particular position in the ground-truth dataset.

4.4.1 Overall Performance

This experiment evaluates the performance of the algorithm for things annotation proposed in Chapter 3. We randomly removed the category tags of a certain percentage, ranging from 10% to 50%, of things from each category of the ground-truth dataset. These things were used to test our approach while the rest were used as the training set. We particularly compared the annotation performance by using i) the features obtained from CNT (i.e., latent features), ii) the features obtained from things’ descriptions (i.e.,
content features), and iii) the combination of the both. Each process was repeated 10 times and the average results were recorded. Similar observations were obtained for different testing percentages. Figure 4.5 shows the result when we removed 30% of things from each category of the ground-truth dataset.

As we mentioned earlier, description of things are normally short and noisy, it is therefore not surprising that the performance based on content features only is worse than the one based on latent features in the most categories. The consistent good performance from the latent features also indicates that our proposed relational network of things is able to capture the correlations among things well. From the figure, we can see that by combining the two together, the performance of all six categories is increased and is the best consistently among the three. It should be noted that for the metrics of hamming-loss, one-error, and coverage, higher value means worse performance. On the other hand, for the average precision metric, higher value means better performance.

4.4.2 Parameters Tuning

This experiment evaluates the impact of tuning parameters γ, ϵ and η in Equation 2.15 on the classification performance using the algorithm of Chapter 3. For the three parameters, they are used to tune the weight of things relevance values calculated from user, time, and location aspects. We conducted experiments using the following four schemes:

- Scheme 1: equal impact (γ=1, ϵ=1 and η=1),
- Scheme 2: bigger impact of user (γ=0.5, ϵ=0.25 and η=0.25),
- Scheme 3: bigger impact of time (γ=0.25, ϵ=0.5 and η=0.25),
- Scheme 4: bigger impact of location (γ=0.25, ϵ=0.25 and η=0.5).
Figure 4.5: Overall performance comparison on six categories

Figure 4.6 shows the result. We can see from the figure that Cooking, Office, and Home categories are not quite sensitive to the user aspect, i.e., the impact of $\gamma$ on classification of these three categories is very limited. The possible reason is that these
Figure 4.6: Parameters tuning on six categories

three categories are already connected by tight relevances for their regular users. As a result, there presents little improvement when increasing user aspect. However, we found that for categories Entertainment, Transportation, and First-Aid, user aspect shows obvious impact since better performance was obtained when $\gamma$ increases. Similarly, Cooking, Office, Home, and Transportation do not show high sensitivity for the time aspect. This is mainly due to the reason that they are already well clustered according to time. On the other hand, for Entertainment and First-Aid, they perform differently when $\epsilon$ increases: the former performs slightly better while the latter gets worse. This should not be surprising since things in the First-Aid category do not possess obvious relevance with time (e.g., it is hard to
find a common time for people to receive initial treatment of injuries or illnesses at work place). Finally, location information related to the prior knowledge on location information (e.g., a user is currently in lab) can help enhance the performance for most categories in our experiments.

4.4.3 Effects on Time and Location Smoothing

Time is continuous. However, in our approach, for the sake of simplicity, we construct the thing-time graph using discrete time information (half-hour in our case). The purpose of this experiment is to investigate the effects of the continuity of time on the performance of our proposed approach.

We propose a method to capture the continuity of time. The approach is based on the intuition that a user who uses a thing at time $t_i$ is likely to use related things around times $t_{i-1}$ and $t_{i+1}$. For example, if a user uses a microwave oven (to heat her lunch) at 12 noon, she is also likely to use the coffee machine before or after 12. Our approach is to add additional edges to the thing-time graph, representing the adjacent times of $t_i$. The number of edges to be added depends on the number of adjacent times, $n_a$. For example, if $n_a$ is 1, two additional edges will be added in the graph from microwave oven to the time nodes of 11am and 1pm in the previous example.

In the experiment, we tested our approach with the smoothing treatment of the thing-time graph ($n_a$ is set as 1). The result is compared with the one from the initial design where no smoothing treatment is received by the graph. Figure 4.7 shows the results. The number in a-axis represents the number of the category (see Table 4.1). From the figure we can see that the performance of all categories except Cooking and First-Aid improved with the smoothed approach. The effects on Cooking and First-Aid are negligible. This is due to the reason that things in Cooking category is already well clustered according to temporal information while things in
First-Aid category do not have obvious relevance with time.

Similar issue exists in location information, which scatters in a discrete spatial dimension. In this experiment, we investigated the impact of clustering locations with prior knowledge of location (e.g., the definition of the lab area) on the classification performance. We propose a method based on the assumption that people in the locations of the same cluster have higher probability to interact with similar things. For each log-in record happened in location \( l_i \), we establish additional \( n \) edges to the other locations in the same cluster that \( l_i \) belongs to. Figure 4.8 shows the result. We can see from the figure that except Transportation, the classification performance of all other categories’ are improved. The possible reason is that the prior location...
knowledge is not very obvious to the category of Transportation.

![Graphs showing impact of clustered location](image)

(a) Hamming Loss  
(b) One-error  
(c) Coverage  
(d) Average Precision

Figure 4.8: Impact of clustered location

4.5 Experiments for T-DisCor+

We have also conducted extensive experiments to validate T-DisCor+ approach. We focus on reporting the evaluation results for things annotations in terms of i) overall performance; ii) impact on introducing spatio-temporal integrity in our approach and iii) spectral clustering performance study.

We randomly removed the category labels of a certain percentage on our groundtruth dataset, ranging from 10% to 50%, of things from each category of the ground-truth
dataset. These things were used to test our approach while the rest were used as the training set. Our algorithm produces a vector of probabilities, representing the assignment probabilities of all labels for a testing object. In our experiments, we ranked these probabilities and chose the top $k$ labels to compare with the ground truth labels. The $k$ value was set to the number of ground truth labels for each testing object and it varies from object to object.

We used the micro-F1 and macro-F1 evaluation measures. The F1 measure is defined as $F_1 = \frac{P \times R}{P + R}$, where $P$ and $R$ are precision and recall respectively. The micro-F1 is defined as:

$$
Micro - F1 = \frac{2 \sum_{j=1}^{c} \sum_{i=1}^{n} \hat{y}_i^j y_i^j}{\sum_{j=1}^{c} \sum_{i=1}^{n} \hat{y}_i^j + \sum_{j=1}^{c} \sum_{i=1}^{n} y_i^j} \quad (4.1)
$$

where $n$ is the number of test data, $y_i$ is the true label vector of the $i$-th sample, $y_i^j = 1$ if the instance belongs to category $j$, $-1$ otherwise. $\hat{y}_i$ is the predicted label vector. The micro-F1 measure weights equally on all samples, thus favoring the performance on common category labels. Macro-F1 is calculated as mean arithmetical value for F1 on each label. It measures weights equally on all the category labels regardless of how many samples belong to it, thus favoring the performance on rare category labels.

### 4.5.1 Overall Performance

This experiment evaluates the performance of the algorithm for things annotation in Chapter 3. We randomly removed the category tags of a certain percentage, ranging from 10% to 50%, of things from each category of the ground-truth dataset. These things were used to test our approach while the rest were used as the training set. We particularly compared the annotation performance by using i) the features obtained from $G$, ii) the features obtained from thing descriptions (i.e., content features $F_C$), and iii) the combination of the both.
Each process was repeated 10 times and the average results were recorded. Similar observations were obtained for different testing percentages. Figure 4.9 shows one result when we removed 30% of things from each category of the ground-truth dataset. As we mentioned earlier, the descriptions of things are normally short and noisy, it is therefore not surprising that the performance based on content features only is worse than the one based on latent features (i.e., $F_L + F_S$) in the most categories. The consistent good performance from the latent features also indicates that our proposed top-$k$ correlation graph of things $G$ is able to capture the correlations among things well. From the figure, we can see that by combining the two together, the performance of all six categories is increased and is the best consistently among the three.

### 4.5.2 Impact of Integrating Spatio-Temporal Information

We argue that user interactions with physical things usually present strong spatial-temporal correlations. In our $T$-DisCor+ approach, we treat the spatial and temporal information of thing usage events inseparable and construct the spatio-temporal graph, which captures periodical patterns between locations and timestamps. We believe that this integration of spatial and temporal information in a single graph can offer bet-
ter performance in discovering correlations of things. To validate this idea, we constructed two independent graphs based on time and location information from thing usage events. Then the random walk with restart was performed in the two graphs separately. Together with the constructed social graph, a relational graph of things was constructed as described in Section 2.4.4 as Equation 2.24. We labeled this approach as No-STI, meaning without spatio-temporal integration and our approach as STI. Similar to the overall performance experiment, we performed things annotation by using features obtained from two different relational graph of things constructed from No-STI approach and our STI approach respectively. Table 4.2 shows the results when we removed 30% of things from each category of the ground-truth dataset.

From the table we can see that, the annotation performance is enhanced for almost all categories by introducing spatio-temporal integrity. The category, Medical/Medicine, is the only one that does not show a distinctive improvement. The reason is that user interactions with things in this category do not have strong connections with spatio-temporal patterns, i.e., people usually do not show periodical patterns when accessing medical related things (e.g., only when they are sick).

Table 4.2: Performance comparison with STI and without STI

<table>
<thead>
<tr>
<th>Category</th>
<th>Entertainment</th>
<th>Office</th>
<th>Transportation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Micro-F1</td>
<td>Macro-F1</td>
<td>Micro-F1</td>
</tr>
<tr>
<td>No-STI</td>
<td>0.6954</td>
<td>0.6533</td>
<td>0.7442</td>
</tr>
<tr>
<td>STI</td>
<td>0.7226</td>
<td>0.6818</td>
<td>0.7854</td>
</tr>
<tr>
<td>Category</td>
<td>Cooking</td>
<td>Medical/Medicine</td>
<td>Home Appliance</td>
</tr>
<tr>
<td></td>
<td>Micro-F1</td>
<td>Macro-F1</td>
<td>Micro-F1</td>
</tr>
<tr>
<td>No-STI</td>
<td>0.7634</td>
<td>0.7213</td>
<td>0.6121</td>
</tr>
<tr>
<td>STI</td>
<td>0.7987</td>
<td>0.7451</td>
<td>0.6162</td>
</tr>
</tbody>
</table>
4.6 Conclusion

In this chapter, we have presented the implementation of our proposed *T-Mine* platform enabling Internet of Things. To validate the feasibility of our proposed things discovery models *T-DisCor* and *T-DisCor+* in Chapter 2, and highlight the benefits of our proposed approaches, things annotation task by exploiting derived things correlations in Chapter 3, we have conducted an extensive set of performance studies and reported our experimental results. We also have introduced a Web-based things management system on top of *T-Mine* platform in Section 4.3.

Recent advances in radio-frequency identification (RFID), wireless sensor networks, and Web services have made it possible to bridge the physical and digital worlds together, where ubiquitous things are becoming an integral part of our daily lives. We view the work presented in this part as a first step towards effective management of things in the emerging Internet of Things era. Other than the contextual information underlying in things usage events, things possess some valuable and prominence attributes which can be utilized and mining to contribute some important mining applications (i.e., things classification etc.) Automatically classifying web objects into manageable semantic categories has long been a fundamental pre-process for indexing, browsing, searching, and mining heterogeneous web objects. In next part, we will explore two significant attributes of things: *semantic labels* and *social tags*, their roles in developing effective things classification approach. Furthermore, we also explore the *availability* of things and present an efficient method to track things availability in real-time.
II

Mining Things and Their Attribute Relationships
Chapter 5

Things Classification Using Bi-Relational Graph of Things and Semantic Labels

Semantic label is an important attribute to describe a thing in terms of functionalities and non-functionalities. For example, we can describe a microwave oven using its semantic labels, e.g., *cooking* (functionality), *Samsung* (non-functionality). Exploiting semantic labels provides a new way in classifying things. In this chapter, we explore the relationship between things and one of their attributes, *semantic labels*.

We construct a general bi-relational mixed graph where the things network and its corresponding semantic label network are connected using label assignments relationship. Under this framework, things classification can be achieved. We propose a graph-based semi-supervised learning approach RMG that leverages random walks on mixed thing-label graphs for effective multi-label classification. We take text objects to engage this approach in this chapter.

In RMG, a mixed graph contains a text-affinity subgraph, a label-correlation subgraph, and text-label edges indicating the assignment relationships. The text-affinity subgraph is constructed using \( \ell_1 \) sparse graph reconstruction in a parameter-free way.
for improved semantic relatedness and robustness against noise, while the label-correlation subgraph captures pairwise correlations among labels. A random walk with restart is then performed on the constructed mixed graph to infer probabilistic assignment relationships between labels and texts.

The main contributions of this chapter are as the following:

- We propose RMG, a graph-based semi-supervised learning framework for multilabel text classification, leveraging random walks on mixed text-label graphs for accurate categorization of textual documents. Our approach exploits the combination of sparse reconstruction with markov chains.

- We adopt the parameter-free $\ell_1$ minimization-based sparse graph reconstruction for generating the text-affinity subgraphs, leading to improved semantic relatedness and robustness against noise. Moreover, the adjacency structure and weight assignment of text affinity subgraph can be obtained with no additional cost using $\ell_1$ based graph reconstruction process.

- We conduct extensive experiments on real Web datasets from Yahoo! in comparison with the state-of-the-art methods, demonstrating the effectiveness of our approach.

### 5.1 Motivations and Challenges

The goal of text classification is to classify textual documents into a certain number of predefined categories, which is an essential technique for handling and organizing textual data. The rapid growth of the World Wide Web in recent years has resulted in the generation of massive textual content of heterogeneous nature. Consequently, automatic text classification has become increasingly important as a key component in
Chapter 5. Things Classification Using Bi-Relational Graph of Things and Semantic Labels

information retrieval, text annotation, and text mining applications [97, 2, 138, 41].

In many real-world applications, textual documents typically can be assigned with multiple categories, and hence have multiple labels. For example, a web page introducing Willard Mitt Romney, the 2012 US presidential candidate, can be labeled as Politics, Celebrities, Feature Story, etc. Thus, text classification generally can be referred to as multi-label categorization, where a document can be assigned one or more (up to \(k\)) category labels drawn from a controlled vocabulary. It should be noted that single-label categorization can be considered as a special case of multi-label categorization.

Effective supervised learning methods for text classification require sufficient amount of labeled data for training. However, labeled data are known to be expensive and time-consuming to obtain in reality [161, 46, 78, 160]. A challenge therefore centers on exploiting unlabeled data to reduce the amount of labeled training data required for effective and efficient multi-label text classification. Surprisingly, although it is important to consider semi-supervised approaches—which utilize both labeled and unlabeled data for training to overcome this challenge—the area of semi-supervised multi-label learning has not been well explored [97, 162, 46]. Only some preliminary efforts have been devoted to semi-supervised multi-label classification, among them are non-negative matrix factorization [78], graph-based methods [145], content-based features [27], and topic models [18, 99].

In the last few years, graph-based approaches have emerged as the most effective for semi-supervised multi-label classification [80, 145, 158, 161]. Such approaches view the whole data set as a graph where the nodes correspond to labeled and unlabeled data points (instances), and the edges reflect the similarities between data points. Many graphs constructed from these approaches are dense graphs that need to be sparsified. Some proposed techniques include sampling nodes or edges [151] and constructing a
new subgraph of the original graph \[77\]. Unfortunately, these methods often require certain expertise and introduce extra computational cost (e.g., inverted index, locality sensitive hashing). In addition, when dealing with multi-label scenarios, many approaches do not consider the interdependencies between labels \([162, 46]\). To overcome these limitations, in this paper we propose RMG (Random Walk on Mixed Graph), a graph-based semi-supervised learning framework for multi-label text classification. RMG fully considers text-text affinity, label-label correlation, together with text-label assignment relationships for effective semi-supervised categorization of texts. In particular, RMG first constructs a mixed text-label graph containing two subgraphs, a text-affinity subgraph and a label-correlation subgraph, as well as edges connecting the texts and labels from the two subgraphs. The internal text-text or label-label edges within the subgraphs capture the affinity or correlation of intra-relationships, and the text-label edges across the subgraphs capture the assignment (category ownership) inter-relationships.

In the text-affinity subgraph, each vertex indicates a textual document (labeled or unlabeled) represented as a feature vector. The edges are weighted reflecting affinity between the involving text pairs. In this study, the text-affinity subgraph is constructed based on the \(\ell_1\) sparse representation, namely \(\ell_1\) graph \([23]\), rather than the traditional one-to-one pairwise similarity graph (e.g., by Gaussian kernel graph construction or \(\epsilon\)-ball construction). Sparse representation involves a one-to-all sparse reconstruction, which is insensitive to data noise and can effectively avoid the propagation of redundant or distracting information \([23, 122]\). This application of sparse reconstruction better captures the semantic relatedness among texts, leading to improved semi-supervised text classification. Furthermore, \(\ell_1\) minimization based graph reconstruction can automatically construct adjacency structure for each vertex and deduce the similarity for pairwise vertexes without assuming the parameters such as the size of neighborhood or threshold of relevance score.
In the label-correlation subgraph, each vertex indicates a category label represented as a binary vector, which has the same size as the total number of texts storing the assignment relationships between the label and each text. Cosine similarity is computed to estimate the correlation between each pair of labels. The mixed graph is constructed by adding text-label assignment edges to the two subgraphs. Then, a random walk with restart (RWR) [120] is performed on the mixed graph to compute text-label relevance, estimating the probabilities of category ownership for each text.

### 5.2 Problem Formulation

Given a set $\mathcal{O}$ of labeled and unlabeled textual documents and a set $\mathcal{L}$ of labels, we aim to automatically assign up to $k$ labels drawn from $\mathcal{L}$ for each unlabeled text. To do so, we compute text-label assignment probabilities by performing a random walk on a mixed text-label graph.

As schematically illustrated in Figure 5.1, in the constructed both text graph and label graph exist as subgraphs, which are connected by an additional bipartite graph induced from label assignments. Consequently, both texts and semantic labels are equally regarded as vertices, and classification problem is transformed to measure how closely a label is related to a text.

The bi-relational graph $G$ mixes two independent subgraphs, namely a text-affinity subgraph $G_\mathcal{O}$ and a label-correlation subgraph $G_\mathcal{L}$, as well as a set of text-label edges, $E(\mathcal{O}\mathcal{L})$, across $G_\mathcal{O}$ and $G_\mathcal{L}$, which indicate the assignment relationships between texts and labels.

**Definition 6** *(Text-Affinity Subgraph)*. Denoted by $G_\mathcal{O} = (V_\mathcal{O}, E_\mathcal{O})$, a text-affinity subgraph is a directed graph, where each vertex in $V_\mathcal{O}$ represents a text in $\mathcal{O}$. Each edge in $E_\mathcal{O}$ is associated with a weight indicating the affinity relationship between the
Definition 7 (Label-Correlation Subgraph). Denoted by $G_L = (V_L, E_L)$, a label-correlation subgraph is an undirected graph, where each vertex in $V_L$ represents a label in $L$. Each edge in $E_L$ is associated with a weight indicating the correlation relationship between the pair of the involving labels.

Figure 5.1: Illustration of a mixed graph. The lower component is the text-affinity subgraph, where the edges indicate text-text affinity. The upper component is the label-correlation subgraph, where the edges indicate label-label correlation. The dashed edges across the two components indicate text-label assignment relationships. For example, text 17 is labeled with Celebrities, Art and Entertainment. We aim to infer the most appropriate labels for the unlabeled texts, which are the ones with question marks in the figure, for example, text 25.
Definition 8 (Mixed Graph). Denoted by $G = (V, E)$, a mixed graph is a directed graph, where the vertex set $V = V_O \times V_L$, the edge set $E = E(O) \cup E(L) \cup E(OL)$. Each edge in $E(OL)$ indicates an assignment relationship between the pair of the corresponding text and the label.

Formally, our multi-label text classification problem can be described as: given a set $O = \{o_1, ..., o_n\}$ of labeled and unlabeled texts and a set of $k$ labels $L = \{l_1, ..., l_k\}$, each text $o_i$ can be extracted as a data point and denoted as feature vector $v_i \in \mathbb{R}^m$, which is associated with a subset of labels $l_i \subseteq L$, where $l_i \in \mathbb{R}^k$ and $l_i^j = 1$ if $o_i$ is associated to $j$-th label, and 0 otherwise. The similarities between texts, measured with $W_O \in \mathbb{R}^{n \times n}$, are induced from the text affinity subgraph $G_O$, where $W_O(i, j)$ denotes the similarity between texts $o_i$ and $o_j$. The correlations between labels, measured with $W_L \in \mathbb{R}^{k \times k}$, are induced from the label correlation subgraph $G_L$, where $W_L(i, j)$ denotes relatedness between labels $l_i$ and $l_j$. Suppose the first $r$ texts are labeled, we aim to predict the most appropriate labels drawn from $L$ for the rest of $n - r$ unlabeled texts.

5.3 Bi-relational Graph Construction

5.3.1 Affinity Subgraph

We begin with constructing the text-affinity subgraph, for which we adopt the $\ell_1$-based one-to-all sparse graph reconstruction to generate the similarity matrix. Compared to traditional one-to-one pairwise similarity graph construction (e.g., Euclidean distance based $\epsilon$-ball neighborhood or kNN [12]), sparse graph reconstruction has several major advantages.

- Sparse graph reconstruction can pass on the valuable information and propagate
discriminative label information for classification because it is able to characterize the local structure and relations [12]. Traditional pairwise similarity calculation for features can be characterized as a one-to-one construction approach, where the generated similarity matrix is dense and cannot keep the locality information such as relationships among neighboring feature vectors that belong to the same category.

- Sparse graph reconstruction is not sensitive to data noise and can effectively avoid the propagation of incorrect information due to the fact that each text only links to a small number of other texts that are semantically related. Data noise abounds especially for Web texts. For instance, social tags of web pages contributed by random users often contain significant redundancy and/or distracting information. Sparse graph representation can effectively remove such noise [23,122].

- Compared with traditional graph construction (i.e., $k$-nearest neighbor and $\epsilon$-ball), $\ell_1$ graph reconstruction does not need to set parameters a-priori (i.e., $k$ in $k$-nearest neighbor or $\epsilon$ in $\epsilon$-ball methods), because $\ell_1$ graph provides a parameter-free graph construction style that can avoid the disadvantages of manual setting of parameters.

- Although Euclidean distance based $\epsilon$-ball neighborhood graph can capture the geometric characteristics, it can not guarantee the connectivity of the constructed graph. In other words, it may produce separated subgraphs. For the $k$-NN method, it can guarantee the connectivity, but can not capture the geometric characteristics among the data points [12]. $\ell_1$ graph can effectively overcome these issues. In the construction process, it simultaneously derives the graph adjacency structure and weights assignment.
Figure 5.2 compares and visualizes traditional pairwise similarity graph construction with $\ell_1$-based sparse graph reconstruction. We randomly selected 100 samples from the Yahoo! Arts dataset and constructed text affinity graphs using both techniques. The visualization clearly shows that sparse reconstruction can remove many redundant links and keep the most related ones. The traditional method produces a link for each pair of texts, and the weight is inversely proportional to the pairwise Euclidean distance in full feature space. Note that we did not set the weight threshold in the visualization. There are edges among each pair of data samples, thus the incorrect information may be propagated among geometrically unrelated data samples (see the left part of Figure 5.2). However, in the constructed sparse graph, only a small number of related samples are selected to have geometrical links. Thus the sparse graph representation can avoid propagation of incorrect information (see the right part of Figure 5.2).

![Figure 5.2: One-to-one pairwise affinity graph vs. sparse graph. We randomly selected 100 vertexes from the Yahoo! Arts dataset and applied both traditional similarity graph construction (left) and sparse graph reconstruction (right). The thickness of lines is proportional to the edge weight.](image)

We describe how to construct the text-affinity subgraph using $\ell_1$-based sparse graph reconstruction in the following. Suppose we have $n$ number of texts, each being rep-
Chapter 5. Things Classification Using Bi-Relational Graph of Things and Semantic Labels

represented as a feature vector of size $m$. Then, the given set of texts can be represented as a set of feature vectors $\mathbf{v} = \{\mathbf{v}_1, ..., \mathbf{v}_n\}$, where $\mathbf{v}_i \in \mathbb{R}^m$. Each feature vector is supposed to be a sparse representation which is formed by all other feature vectors in the dataset with nonzero coefficients [34]. The sparse representation demonstrates the relationship between each feature vector and other feature vectors, which is used to construct the text-affinity graph.

Although finding the sparsest solution for reconstruction is an NP-hard problem due to the nature of the underlying combinatorial optimization, the sparse representation still can be recovered by convex $\ell_1$-norm minimization according to [32]. Thus, given a text $\mathbf{v}_i$, its relationship with other texts can be obtained by $\mathbf{v}_i = \mathbf{V}^i \mathbf{w}$, where $\mathbf{v}_i \in \mathbb{R}^m$ is the sample to be reconstructed, $\mathbf{w} \in \mathbb{R}^n$ is the reconstruction coefficients, $\mathbf{V}^i = [\mathbf{v}_1, ..., \mathbf{v}_{i-1}, \mathbf{v}_{i+1}, ..., \mathbf{v}_n] \in \mathbb{R}^{m \times (n-1)}$, which is formed by other texts in the dataset except for text $\mathbf{v}_i$. This equation can be solved by:

$$
\min_{\mathbf{w}} ||\mathbf{w}||_1, \quad s.t. \quad \mathbf{v}_i = \mathbf{V}^i \mathbf{w}
$$

(5.1)

where $|| \cdot ||_1$ is the $\ell_1$ norm, tending to minimize the $\ell_1$ norm of reconstruction error. To solve this equation using linear programming algorithms, it requires that $\mathbf{v}_i = \mathbf{V}^i \mathbf{w}$ must be an under-determined system of linear equations. For example, the feature dimensionality must be much smaller than the size of the sample space, i.e., $m << n$ [21]. However the datasets in our experiments do not meet this prerequisite. For example, the Yahoo! Arts dataset contains about 5,000 samples and the feature dimensionality is over 20,000 ($m >> n$). To tackle this problem, we can either reduce the feature dimensionality or augment $\mathbf{V}^i$ by an $m \times m$ identity matrix [130] to turn this over-determined system into an under-determined system. Thus, Equation 5.1 can be transformed to the following:

$$
\min_{\mathbf{w}} ||\mathbf{w}||_1, \quad s.t. \quad \mathbf{v}_i = \mathbf{B} \mathbf{w}
$$

(5.2)
where $B = [V^i|I] \in \mathbb{R}^{n \times (m + n - 1)}$. Considering the existence of feature noise $\xi$ in $v_i = V^i w + \xi$, where $\xi \in \mathbb{R}^m$, the final equation can be formulated as:

$$\min_{\hat{w}} \|\hat{w}\|_1, \quad s.t. \quad v_i = B \hat{w} \quad (5.3)$$

where $\hat{w} = [w^T | \xi^T]^T$.

This optimization problem is convex and there exists a global optimal solution, which can be solved in polynomial time using general $\ell_1$ norm optimization programming packages and toolboxes such as SparseLab$^1$ and $\ell_1$MAGIC$^2$. After obtaining the reconstruction coefficients $w$ by solving Equation 5.3, we can utilize them as weights between text $v_i$ and other samples. This is because the reconstruction coefficients characterize the natural relationship among the text samples and indicate the affinity relationship between $v_i$ and other samples in the dataset. Moreover, the sparse representation is automatically conducted by solving Equation 5.3, which indicates the adjacency structure of graph. As a result, we can do the one-to-all sparse reconstruction for each text in the dataset to finally obtain the text graph $G_O$. We use the absolute value of coefficients $|\hat{w}_i|$ as the weight in the $\ell_1$ based text affinity subgraph. The weight matrix is $W_O$, where each entry $W_O(i, j)$ indicates the edge weight for texts $i$ and text $j$. It is worth mentioning that the text affinity subgraph constructed from $\ell_1$ graph is a directed graph.

Algorithm 3 summarizes the $\ell_1$-based sparse graph reconstruction method for constructing text-affinity subgraphs.

---

$^1$http://sparselab.stanford.edu
$^2$http://users.ece.gatech.edu/~justin/l1magic/
Algorithm 3: $\ell_1$-based text-affinity subgraph construction

\begin{algorithm}
\caption{$\ell_1$-based text-affinity subgraph construction}
\begin{algorithmic}[1]
\Input a collection of feature vectors of texts $v = \{v_1, \ldots, v_n\}, v_i \in \mathbb{R}^m$
\Output similarity matrix $W_O \in \mathbb{R}^{n \times n}$
\For{$i = 1: n$}
\State $B_i = [v_i | I]$;
\State Solving $\ell_1$ minimization: $\hat{w}_i = \arg\min_{w} \|w_i\|_1$, s.t., $v_i = B_i \hat{w}_i$;
\For{$j = 1: n - 1$}
\If{$i > j$}
\State $W_O(i, j) = |\hat{w}_i^j|$
\Else
\State $W_O(i, j) = |\hat{w}_i^{j-1}|$
\EndIf
\EndFor
\EndFor
\end{algorithmic}
\end{algorithm}

5.3.2 Categorical Correlation Graph

The label-correlation subgraph captures the pairwise correlation relationships among category labels. For example, a text belonging to the category *politics* (see the example in the introduction) is more likely to be related to the label *presidential campaign* rather than *cooking*.

Let $G_L = (V_L, E_L)$ denote the label-correlation subgraph, where each vertex represents a category label. We represent a vertex as a binary vector $l_i \in \{0, 1\}^k$, where $k$ is the number of labels. $l_i(n) = 1$ means that text $o_i$ is assigned with the $n$-th label, and 0 otherwise. Thus the weight matrix for the subgraph can be formulated as:

$$W_L(i, j) = e^{(-\lambda(1-\cos(l_i, l_j)))}$$ (5.4)

where $\lambda$ is the hyper parameter and $l_j$ is the binary vector whose entries are set to 1 when the corresponding text is assigned with the $i$-th label in the ground truth dataset, and 0 otherwise. $\cos(l_i, l_j)$ is the cosine similarity between $l_i$ and $l_j$, which is calcu-
lated as the following:

$$\cos(l_i, l_j) = \frac{\langle l_i, l_j \rangle}{||l_i|| ||l_j||} \quad \text{(5.5)}$$

Once we obtain the text-affinity subgraph $G_C$ and the label-correlation subgraph $G_L$, we can complete the construction of the mixed graph $G$ by combining the two subgraphs and adding the text-label edges across the subgraphs based on label assignment in training samples. For example, in Figure 8.1, text 17 in the label-correlation subgraph has assigned three labels, i.e., Celebrities, Art, and Entertainment in the text-affinity subgraph. In this case, three text-label edges are added in the mixed graph linking them together.

Instead of weighting these three labels equally, we introduce an adaptive weight calculation, which assigns different weight to edges linking text and their corresponding labels. The edge weight $W_{OL}(o_t, l_i)$ from a text $o_t$ to a label $l_i$ can be derived from $o_t$’s neighbouring nodes and calculated as follows:

$$W_{OL}(o_t, l_i) = \frac{\sum_{o_{t'} \in \Omega_{o_t}} \mathbb{I}(o_{t'}, l_i)}{\sum_{l_{i'} \in \Omega_{l_i}} \sum_{o_{t'} \in \Omega_{o_t}} \mathbb{I}(o_{t'}, l_{i'})} \quad \text{(5.6)}$$

where $\Omega_{o_t}^L$ denotes the label set of text $o_t$, $\Omega_{o_t}$ denotes the neighbour nodes connecting to text $o_t$. $\mathbb{I}(o_{t'}, l_i)$ is 1 if text $o_{t'}$ has label $l_i$, 0 otherwise. This way in calculating weight reflects the more the number of neighbouring nodes share specific label in label set of text $o_t$, the more important the specific label is for the text $o_t$.

### 5.4 Learning Algorithm

After the mixed graph $G$ is constructed, random walk with restart (RWR) [132] is applied on $G$ to compute the text-label relevance for each unlabeled text. High relevance indicates high probability of correct assignment.
Chapter 5. Things Classification Using Bi-Relational Graph of Things and Semantic Labels 87

Firstly, we need to obtain the affinity matrix from $G$:

$$
W = \begin{bmatrix}
W_O & W_{OL} \\
W_{LO} & W_L \\
\end{bmatrix}
$$

(5.7)

where $W_O$ is calculated from Algorithm 3, $W_L$ is calculated from Equation 5.4. $W_{OL}$ and $W_{LO}$ are binary vectors indicating the category label ownership of each text. If text $o_i$ belongs to category $l_j$, $W_{OL}(i, j) = 1$. In addition, $W_{LO} = W_{OL}^T$.

The corresponding transition probability matrix $P$ stemmed from the weight matrix $W$ can be denoted as:

$$
P = \begin{bmatrix}
P_{OO} & P_{OL} \\
P_{LO} & P_{LL} \\
\end{bmatrix}
$$

(5.8)

where $P_{OO}$ and $P_{LL}$ are the intra-transition probability matrices within the text-affinity subgraph $G_O$ and the label-correlation subgraph $G_L$ respectively. $P_{OL}$ and $P_{LO}$ are the inter-transition probability matrices between $G_O$ and $G_L$.

Then, we set a jumping probability $\alpha \in [0, 1]$, which is the probability that the random walker jumps from one subgraph to the other. Not all the vertexes in $G$ are connected with text-label edges. For example, the unlabeled texts are not assigned any labels yet and they are not connected with text-label edges. $d_i^{OL} \geq 0$ means that text $o_i$ is connected with at least one text-label edges. During a random walk, if the walker is currently on a text vertex in $G_O$ that is connected with at least one text-label edge, she will jump to the label-correlation subgraph $G_L$ with $\alpha$ probability, or stay in $G_O$ with $(1 - \alpha)$ probability. Now we explain the details on transition matrix using the following equations:

- $P_{OO}(i, j)$ is proportional to matrix $W_O$ obtained from Algorithm 3.

$$
P_{OO}(i, j) = Pr(o_j|o_i) = \begin{cases}
\frac{W_O(i, j)}{d_i^O} & \text{if } d_i^{OL} = 0 \\
(1 - \alpha)\frac{W_O(i, j)}{d_i^O} & \text{otherwise}
\end{cases}
$$

(5.9)
Since $W_O$ measures the text-text similarity, $d_i^O = \sum_j W_O(i, j)$ denotes the sum of edge weights for vertex $i$ and $j$ on the text-affinity subgraph.

- $P_{\mathcal{L}\mathcal{L}}(i, j)$ is proportional to the correlation matrix $W_L$ calculated from Equation 5.5.

\[
P_{\mathcal{L}\mathcal{L}}(i, j) = Pr(l_j | l_i) = \begin{cases} 
\frac{W_L(i, j)}{d_i^L} & \text{if } d_i^O \geq 0 \\
(1 - \alpha) \frac{W_L(i, j)}{d_i^L} & \text{otherwise} 
\end{cases} \tag{5.10}
\]

- $P_{O\mathcal{L}}(i, j)$ is the transition probability matrix from texts to labels, and is proportional to $W_{O\mathcal{L}}$.

\[
P_{O\mathcal{L}}(i, j) = Pr(l_j | o_i) = \begin{cases} 
\alpha \frac{W_{O\mathcal{L}}(i, j)}{d_i^O} & \text{if } d_i^O \geq 0 \\
0 & \text{otherwise} 
\end{cases} \tag{5.11}
\]

where $d_i^O$ is the total text-label edge weights of vertex $i$ in the label-correlation subgraph, i.e., $d_i^O = \sum_j W_{O\mathcal{L}}(i, j)$.

- $P_{\mathcal{L}O}(i, j)$ is the transition probability matrix from labels to texts, and is equal to the transpose of $P_{O\mathcal{L}}(i, j)$.

\[
P_{\mathcal{L}O}(i, j) = P_{O\mathcal{L}}^T(i, j) = Pr(o_j | l_i) = \begin{cases} 
(1 - \alpha) \frac{W_{O\mathcal{L}}^T(i, j)}{(d_i^O)^T} & \text{if } d_i^O \geq 0 \\
0 & \text{otherwise} 
\end{cases} \tag{5.12}
\]

where $d_i^O$ is the total text-label edge weights of vertex $i$ in the label-correlation subgraph, i.e., $d_i^O = \sum_j W_{O\mathcal{L}}(i, j)$.

The transition probability matrix can be rewritten as follows:

\[
P = \begin{bmatrix} (1 - \alpha)W_O^{-1} & \alpha W_{O\mathcal{L}}^{-1} \\
\alpha W_{O\mathcal{L}}^{-1}(D_{O\mathcal{L}})^{-1} & (1 - \alpha)W_{\mathcal{L}}^{-1} \end{bmatrix} \tag{5.13}
\]
where $D_O$, $D_L$, $D_{OL}$, $D_{OL^T}$ are diagonal matrices and the $i$-th entry in diagonal of $D$ is its sum of edge weights.

\[
\begin{align*}
D_O &= \text{diag} \left[ d_{O1}, \ldots, d_{On} \right] \\
D_L &= \text{diag} \left[ d_{L1}, \ldots, d_{Lm} \right] \\
D_{OL} &= \text{diag} \left[ d_{OL1}, \ldots, d_{OLm} \right] \\
D_{LO} &= \text{diag} \left[ d_{OL^T1}, \ldots, d_{OL^Tn} \right]
\end{align*}
\] (5.14)

At this point, we conduct a random walk with restart (RWR) on the mixed graph $G$ to estimate how probable an unlabeled text belongs to the category labels. RWR provides a good relevance score between two nodes in a graph, and has been successfully used in many applications such as automatic image captioning [93], recommender systems [39], and link prediction [72]. The goal of using RWR in our work is to find the category labels that have top-$k$ highest relevance scores for a given unlabeled text.

Briefly, RWR works as the following. A random walker on a vertex $i$ traverses randomly along its edges to the neighboring vertexes based on the transition probability (Equation 5.13) with the probability of $1 - \alpha$, or jumps back to vertex $i$ with the probability of $\alpha$. Let $u_i(j)$ denote the steady state visit rate of vertex $j$ starting from vertex $i$, which can be used to estimate the affinity between vertex $i$ and $j$. Below, we describe how RWR is applied in the computation of relevance score.

Given an unlabeled testing text $o_i$, its steady state probability can be obtained by using $u_o = (1 - \alpha)P^Tu_o + \alpha v_o = \alpha(I - (1 - \alpha)P)^{-1}v_o$, where $v_o$ is the restart vector. We initialize $v_o$ as a 0 vector except the $i$-th entry (set to 1). We then execute the RWR process till convergence. The converged probability vector $u_o$ represents the steady state visit rates of all vertexes in $V(O)$ and $V(L)$ on the mixed graph $G$ starting from $o_i$. From $u_o$, we are interested in the visit rates of vertexes in $V(L)$, which can be used to estimate the assignment probabilities of those labels to unlabeled text $o_i$. 
Chapter 5. Things Classification Using Bi-Relational Graph of Things and Semantic Labels

We adopted the fast RWR proposed in [120], which has been widely used in the community. This fast RWR takes advantage of low rank approximation and graph partition. The weight matrix $\tilde{W}$ of the mixed graph $G$ is obtained from normalized graph Laplacian. We implemented an efficient NB_Lin RWR algorithm, summarized in Algorithm 4, which assumes that the partition number is equal to the number of vertexes in matrix $\tilde{W}$. There are two main phases in this algorithm. The first one is the pre-computational phase (lines 2 and 3), which calculates low rank approximation for the weight matrix $\tilde{W}$ and matrix inversion $\Lambda \in \mathbb{R}^{(k \times k)} (k < n)$. The second one is the online-query phase (line 4), which computes the ranked convergence assignment probability vector for each testing sample with reduced number of matrix-vector multiplications. It should be noted that $U$ denotes the $n \times t$ node-concept matrix, $S$ denotes the $t \times t$ concept-concept matrix, and $V$ the $t \times n$ denotes concept-node matrix. Interested readers are referred to [120] for more details of the algorithm.

**Algorithm 4: NB_Lin based RWR**

**Input:** normalized weight matrix $\tilde{W}$ associated with $W$; a set of testing text vectors $\alpha = \{o_1, ..., o_t\}$

**Output:** ranked convergence assignment probability vectors for $\alpha$

```
1 for i = 1; i <= t do
2   Low rank approximation $\tilde{W} = USV$;
3   Precompute and store $\Lambda = (S^{-1} - \alpha VU)^{-1}$;
4   Compute and output $u_{oi} = (1 - \alpha)(v_{oi} + \alpha U \land Vv_{oi})$;
5 end
```

5.5 Algorithm Analysis

This section offers some brief discussions on how to improve the computation efficiency of our approach. In this work, we adopt the $\ell_1$ graph for texts subgraph construction. Given a collection of data points (i.e., each text can be considered as a virtual data point), $\ell_1$-graph builds the similarity graph by finding the sparse representation for
each text using the representation coefficient between the text and the other texts in the dataset. Although the sparse similarity graph performs well, it needs an iterative optimization process for respective objective function, and does not have a closed form like $\ell_2$ norm. As a result, the computational complexity might increase. We consider some possible strategies to improve the efficiency of our approach.

- Reducing the search space: the sparse graph reconstruction process (line 4 in Algorithm 3) is a one-to-all process that computes $v_i$’s coefficient with other $n - 1$ samples in the dataset, which means the size of the search space is $n - 1$. We could run the $k$-NN method to reduce the search space by finding the $k$ nearest neighbors for each data point, before performing the $\ell_1$ minimization process. Consequently, we only need to do a one-to-$k$ reconstruction rather than the one-to-all reconstruction.

- Reducing the dimension of feature vector: we also could reduce the size of the original feature vector\(^3\) by adopting some dimensionality reduction methods such as Principal Component Analysis (PCA). PCA is an unsupervised approach, which is widely used in the preprocessing stage for dimension reduction. Given a $n \times m$ text-term matrix (the number of texts and terms are $n$ and $m$ respectively), PCA uses the $k$-leading eigenvectors of the $n \times n$ covariance matrix to trim the matrix to a lower $k$-dimensional space. These leading eigenvectors correspond to the linear combinations of the original variables that account for the largest amount of term variability.

The basic process can be described as follows:

1. Constructing the document-term matrix $M \in \mathbb{R}^{n \times m}$, $n$ is the number of documents, and $m$ is the size of dimension for the feature vector.

\(^3\)The dimensions of original text features are usually quite large.
2. Calculating the covariance matrix $C \in \mathbb{R}^{n \times n}$ and computing all $n$ eigen-values and corresponding eigenvectors for $C$, and taking out the most $k$ highest eigenvalues and corresponding eigenvectors. So that, using the $k$ eigenvectors form a new matrix $C' \in \mathbb{R}^{n \times k}$, where $k \ll m$.

3. Multiplying $C'$ with $U^T$, where $C = UU^T$ to obtain the base matrix $B \in \mathbb{R}^{m \times k}$.

4. Dimensionality reduction for each original data sample: for each original data sample $v_i \in \mathbb{R}^{1 \times m}$, it multiplies $B \in \mathbb{R}^{m \times k}$, so that, we can obtain the reduced new feature vector for each data sample $v'_i \in \mathbb{R}^{1 \times k}$. As a result, the dimensionality of each data feature vector is reduced from $m$ to $k$-dimension ($k \ll m$).

Both of these considerations are related to some parameters tuning process for discovering the most suitable parameters (e.g., $k$), which are the part of our future work.

5.6 Experiments

We have conducted extensive experiments to validate our proposed semi-supervised learning approach for text classification using real datasets from the World Wide Web. In this section, we describe the experimental settings, introduce the comparison partners, and report the experimental results.

5.6.1 Experimental Settings

We used the Yahoo! datasets [126] to evaluate our approach, for convenience, we name our approach as RMG (Random Walk on Mixed Graph). The datasets consist of six datasets that are web pages collected through the hyperlinks from the top directory.
Chapter 5. Things Classification Using Bi-Relational Graph of Things and Semantic Labels

The six datasets used in our experiments are: Arts, Business, Computers, Education, Entertainment, and Health. To conduct the experiments, we randomly selected 3,000 samples from each dataset, removed the category labels for 1,500 samples and used them as the testing texts. We aimed to investigate whether our proposed approach can recover the category labels for these testing texts. All experiments were executed on a PC with Intel i7 2.7GHz CPU and 8GB memory. For better performance, we transformed the features vectors into tf-idf format, and then normalized each tf-idf vector using \( \hat{v} = \frac{\vec{v}}{||\vec{v}||_2} \). Table 5.1 shows the statistics of the six datasets used in the experiments.

Table 5.1: Yahoo! datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th># features</th>
<th># labels</th>
<th>Ave. # labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arts</td>
<td>23,146</td>
<td>26</td>
<td>3.33</td>
</tr>
<tr>
<td>Business</td>
<td>21,924</td>
<td>30</td>
<td>4.76</td>
</tr>
<tr>
<td>Computers</td>
<td>34,096</td>
<td>33</td>
<td>5.14</td>
</tr>
<tr>
<td>Education</td>
<td>27,534</td>
<td>33</td>
<td>4.94</td>
</tr>
<tr>
<td>Entertainment</td>
<td>32,001</td>
<td>21</td>
<td>5.15</td>
</tr>
<tr>
<td>Health</td>
<td>30,605</td>
<td>32</td>
<td>4.11</td>
</tr>
</tbody>
</table>

RMG produces a vector of probabilities, representing the assignment probabilities of all category labels for a testing text. In our experiments, we ranked these probabilities and chose the top \( k \) labels to compare with the ground truth labels. The \( k \) value was set to the number of ground truth labels and it varies from text to text. We used the micro-F1 and macro-F1 evaluation measures. The F1 measure is defined as

\[
F_1 = 2 \frac{P \times R}{P + R},
\]

where \( P \) and \( R \) stand for precision and recall respectively. The micro-F1 is defined as:

\[
Micro - F1 = \frac{2 \sum_{j=1}^{c} \sum_{i=1}^{n} \tilde{y}_{i}^{j} y_{i}^{c}}{\sum_{j=1}^{c} \sum_{i=1}^{n} \tilde{y}_{i}^{j} + \sum_{j=1}^{c} \sum_{i=1}^{n} y_{i}^{c}}
\]

\(^{4}\)http://cs.adelaide.edu.au/~lina/Data/yahoo.rar
where \( n \) is the number of test data, \( y_i \) is the true label vector of the \( i \)-th sample, \( y_i^j = 1 \) if the instance belongs to category \( j \), \(-1\) otherwise. \( \hat{y}_i \) is the predicted label vector. The micro-F1 measure weights equally on all samples, thus favoring the performance on common category labels. Macro-F1 is calculated as mean arithmetical value for F1 on each label. It measures weights equally on all the category labels regardless of how many samples belong to it, thus favoring the performance on rare category labels [136].

### Table 5.2: Overall performance comparison

<table>
<thead>
<tr>
<th>Methods</th>
<th>1%</th>
<th>5%</th>
<th>10%</th>
<th>20%</th>
<th>30%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metrics</td>
<td>MicroF1</td>
<td>MacroF1</td>
<td>MicroF1</td>
<td>MacroF1</td>
<td>MicroF1</td>
</tr>
<tr>
<td>SVM</td>
<td>0.4232</td>
<td>0.3929</td>
<td>0.4672</td>
<td>0.4586</td>
<td>0.5905</td>
</tr>
<tr>
<td>GH</td>
<td>0.4877</td>
<td>0.4757</td>
<td>0.5334</td>
<td>0.5202</td>
<td>0.6427</td>
</tr>
<tr>
<td>wvRN</td>
<td>0.4308</td>
<td>0.4135</td>
<td>0.4858</td>
<td>0.4793</td>
<td>0.6071</td>
</tr>
<tr>
<td>wvRN+RL</td>
<td>0.4417</td>
<td>0.4244</td>
<td>0.5077</td>
<td>0.4914</td>
<td>0.6233</td>
</tr>
<tr>
<td>RMG</td>
<td>0.5520</td>
<td>0.5239</td>
<td>0.6514</td>
<td>0.6230</td>
<td>0.6872</td>
</tr>
</tbody>
</table>

#### 5.6.2 Comparison Partners

We compared RMG with the following four state-of-the-art text classification methods.

- **Gaussian Field and Harmonic Function (GH).** This method is based on a Gaussian random field model from a weighted graph, where labeled and unlabeled data are represented as vertexes and their pairwise similarities as edge weights [161].

- **Weighted-vote Relational Neighbor Classifier (wvRN).** This method predicts the category labels \( l_{o_i} \) for text \( o_i \) based on the weighted mean of its neigh-
bors [84].

\[
P_{r}(l_{o_{i}|N_{o_{i}}}) = \frac{1}{Z} \sum_{o_{j} \in N_{o_{i}}} r_{o_{i}o_{j}} P_{r}(l_{o_{j}|N_{o_{j}}}) \tag{5.16}
\]

where \( Z \) is the normalized factor, \( N_{o_{i}} \) is the neighborhood of instance \( o_{i} \), and \( r_{ij} \) is the edge weight between \( o_{i} \) and \( o_{j} \). In this case, wvRN does not use collective classification. Therefore, any neighbors without labels are ignored.

- Relaxation labeling (RL) + wvRN. This method performs collective classification by using relaxation labeling techniques. The label assignment probability of \( o_{i} \) is calculated by considering both the weighted mean of the assignment probabilities of a text in its neighborhood, and the current assignment probability \( o_{i} \) [83].

\[
P_{r}^{(t+1)}(l_{o_{i}|N_{o_{i}}}) = a^{(t+1)} \frac{1}{Z} \sum_{o_{j} \in N_{o_{i}}} r_{ij} P_{r}^{t}(l_{o_{j}|N_{o_{j}}}) + (1 - a^{(t+1)}) P_{r}^{t}(l_{i}|N_{i}) \tag{5.17}
\]

where a bigger \( a \) means that the assignment probability prediction for a given text is more affected by its close-by neighbors. In our experiments, we set \( a = 0.5 \).

- SVM. We also used the support vector machine (SVM) as a comparison partner. This method decomposes the multi-label classification problem into multiple binary classifications. The one-versus-all classification process is conducted for each label, where the texts from the category of interest are labeled as positive, and the rest are labeled as negative.
5.6.3 Experimental Results

In this section, we report two experimental results. The first set of experiments is to compare the overall performance of our approach with the four baseline methods mentioned in Section 5.6.2. The second experiment is to demonstrate the feasibility of using $\ell_1$ graph to estimate the similarity of texts.

5.6.3.1 Overall performance comparison

Here we report the averaged performance comparison over all of the six Yahoo! datasets for varying training ratios (percentage of labeled data used for training). The results are based on 5-fold cross validation.

In this series of experiments, we used the following training ratios: 1%, 5%, 10%, 20%, and 30%. The experimental results are shown in Table 5.2 and are also visualized in Figure 5.3. From the results we can observe that RMG clearly outperforms the other comparison partners in all cases. We can also observe that the performance of semi-supervised methods (e.g., RMG and GH) are not as sensitive as that of supervised methods (SVM) when the training ratio increases. This is because semi-supervised learning methods already make use of the information in unlabeled data, even though the percentage of labeled data is small.

We also conducted experiments on individual data set to compare the performance of RMG and other four classification methods. Table 5.3, which is also visualized in Figure 5.4, shows the experimental results (based on 5-fold cross validation) when the training ratio is fixed at 10%. It should be noted that other training ratios gave the similar results. From the results we can observe that RMG consistently outperforms all other comparison partners on each of the six datasets.
5.6.3.2 Comparing subgraph construction methods

In this series of experiments, we compared different ways of constructing the text-affinity subgraph, i.e., the traditional pairwise similarity graph construction method using the Gaussian Kernel function and our $\ell_1$-based sparse graph reconstruction. Traditional graph construction consists of two steps:

- Graph adjacency construction: This includes two basic methods, $\epsilon$-ball and $k$-nearest neighbor [12]. In this paper, we connected each text to its $k$-nearest neighbors. We set $k = 10$ in the experiments.
Chapter 5. Things Classification Using Bi-Relational Graph of Things and Semantic Labels

- Assign weights to the edges: we used Gaussian kernel function to calculate edges’ weight. The traditional method assigns symmetric non-negative edge weights $W_O(i, j)$ using Gaussian Kernel function as follows.

$$W_O(i, j) = \exp\left(\frac{||v_i - v_j||^2}{\sigma^2}\right)$$  \hspace{1cm} (5.18)

where $\exp(\cdot)$ is the exponential function, $|| \cdot ||$ is the Euclidean norm for $o_i$ and $o_j$, and $\sigma$ is the kernel parameter determining the geometrical structure of the mapped data in the kernel space. A small $\sigma$ value would lead to large inter-category or intra-category variations for all the mapped data. A big $\sigma$ value would lead to convergence of mapped data to a single point. Based on these observations and following the suggestions in [128], we set $\sigma = \frac{\sum_{i,j} ||v_i - v_j||}{n(n-1)}$.

The experimental results are shown in Figure 5.5. From the results we can observe that sparse graph reconstruction leads to more robust performance than the Gaussian kernel method. These experimental results validate our $\ell_1$-based sparse graph reconstruction approach, which are also consistent with other applications of the methods such as image processing [128].
Chapter 5. Things Classification Using Bi-Relational Graph of Things and Semantic Labels

5.7 Related Work

Over the last few years, multi-label text classification has gained a significant momentum in both research and development [42, 58, 146, 152, 160]. Existing multi-label text classification methods generally fall into two categories: supervised or semi-supervised. For supervised multi-label learning, most existing work attack this problem by exploiting a binary classification approach, which decomposes multi-label classification into multiple independent binary classification problems, one for each category [58, 136]. Such an approach generally suffers from a number of disadvantages. Firstly, it does not scale to large number of categories since a binary classifier has to be built for each category. Secondly, it treats each category independently, which is unsatisfactory since categories are usually not independent, instead, most of them exhibit strong correlations.

Recently, semi-supervised learning has become increasingly prominent for multi-label text classification to overcome problems in supervised learning. Compared to supervised methods, semi-supervised approaches effectively make use of the information provided by unlabeled data [78]. They have proved to be superior especially when
the amount of training data is relatively small. Zhu, Ghahramani, and Lafferty [161] treated the learning problem as a Gaussian random field on a weighted graph of labeled and unlabeled objects. Zhou et al. [158] proposed a semi-supervised learning framework to achieve local and global consistency. Jensen, Neville, and Gallagher [56] proposed a collective classification inference approach that explores the relevance among labels. A relational classifier is constructed based on the relational features of labeled samples, and an iterative process is implemented to learn class labels for the unlabeled samples [85, 102]. Ghamrawi and McCallum [42] proposed a conditional random field (CRF)-based model to parameterize label concurrence. Chen et al. [19] further considered label dependency, which is treated as a new constraint in the model to formulate an optimized framework. Very recently, Ramage et al. [99] exploited correlations between labels and their topics based on a latent topic model. Yin et al. [145] formulated classification as an optimization problem on a mixed graph containing Web objects and social labels. Zhang [152] proposed to use Bayesian Networks to reveal label dependencies. Liu et al. [78] proposed a semi-supervised multi-label learning method based on constrained nonnegative matrix factorization, which exploits unlabeled data as well as category correlations. Unfortunately, all these approaches introduce additional computation cost to some extent.

The current trend for improving the accuracy of the multi-label classification and label propagation is to consider the correlations among categories. Yu et al. [146] proposed a multi-label informed latent semantic indexing (MLSI) that preserves feature information as well as captures label correlations. Zhu et al. [160] proposed a maximum entropy-based method that explicitly models mutual correlations among classes by constructing a conditional probability model from the training data. Ueda and Saito [125] proposed a generative model that incorporates pairwise correlations between categories into multi-label learning.
Our approach constructs a mixed graph containing texts and their labels, and treats them as homogeneous in the same graph. We exploit random walk with restart (RWR) to capture their relationships on this mixed graph. RWR is widely used in network analysis, link prediction [8], semantic annotation and estimation of clustering coefficients in social networks [143, 49], image annotation [128], correlation detection for multimedia resources [94], and very recently latent relevance discovery in Web of Things [138]. The work presented in [128] is the one close to our approach. The authors proposed a framework to leverage RWR on bi-relational graphs for images annotation. However, in our work, unlike traditional one-to-one graph construction (e.g., Gaussian Kernel or Heat Kernel) to calculate the similarity between objects in graph-based semi-supervised learning, e.g., [161, 158], we adopt $\ell_1$ based graph to uncover the relationships among texts, which offers improved semantic relatedness and robustness against noise, as demonstrated in our experiments.

Finally, sparse graphs have been used in various interesting applications such as image analysis [75], unsupervised dimension reduction [98], music structure analysis [65], and subspace learning [34]. To the best of our knowledge, we are the first to apply sparse reconstruction in multi-label text classification.

5.8 Conclusion

In this chapter, we have proposed RMG, a semi-supervised approach for multi-label classification by leveraging random walks on thing-label mixed graph. A mixed graph contains a thing-affinity subgraph, a label-correlation subgraph, and thing-label edges across the subgraphs indicating the assignment relationships. We take text objects as representations, and exploit the $\ell_1$-based sparse graph reconstruction to construct the text-affinity subgraph for improved relatedness and robustness against noise. Random walk with restart is used on the constructed mixed graph to estimate the text-label as-
assignment probabilities for text classification. Extensive experiments on real datasets from Yahoo! have demonstrated the effectiveness of our approach, compared to existing text classification methods. In next chapter, another kind of descriptive attribute of things, social tags, will be studied.
Chapter 6

A Discriminative Model for Things Classification by Learning Social Tags

Internet of Things assimilate physical world and virtual world, where interactions between things and people make the dominant contribution. One of major human behaviors on things is that human tend to mark their interested things as personalized footprint. More and more web objects (i.e., things) are increasingly annotated with human interpretable labels (i.e., social tags), which can be considered as an auxiliary attribute and side information to assist the object classification.

In this chapter, we study web object classification problem with the novel exploration of its social tags. We propose a unified, discriminative tag-centric model for web object classification by jointly modeling the objects category labels and their corresponding social tags and uncovering the relevance among social tags.

6.1 Problem Formulation

Classification of web objects, however, is a much more challenging task due to the specific characteristics of the data [145]. Unlike traditional text documents, web objects cannot be easily extracted with meaningful features. Web objects often exist in isolate settings with little interconnections. Finally, it is usually laborious, sometimes
even infeasible, to create training data for classification due to high diversity of web objects.

Recent research on social tagging systems has been motivated by a vision that the tagging data produced by Web users can be used for social classification, i.e., the collective classification of resources into a commonly agreed structure [163]. According to Strohmaier et al. [115], users’ tagging behaviors can be grouped into two main motivations: Categorizers where users use tags to categorize objects and Describers where users use tags to describe objects. These tags reflect, explicitly or implicitly, the semantics of web objects from users’ point of view and reveal the information about an object such as which category it belongs to or what it looks like. This makes tags an ideal type of data for web objects classification. We categorize social tags’ roles in our work from two aspects:

- **Social Tags as Puzzles**: On the one hand, each social tag is like a puzzle, only conveying a piece of information of a web object. By combining multiple tags of the object, it is possible to obtain more information of the object. For example, the tag “lady gaga” may associate with a news, a Lady Gaga album or photo (see Figure 6.1). However, if we combine the information revealed by the top 5 tags together, the web object is obviously Lady Gaga’s Fame Monster album, which could be assigned to the “Music” category.

- **Social Tags as Bridge**: On the other hand, users tend to tag different types of objects using similar vocabularies. Heterogeneous web objects are usually connected through common tags. For example, in Figure 6.2, web page, YouTube clip, the Fame Monster album are linked by the tag “lady gaga”. If the web page has a category of “Music”, the album and the clip are likely to be labeled with the same category. It is clear that social tags act as a bridge, to transfer category information between different domains [145].
Figure 6.1: Tags as Puzzles: top-5 association tags assigned by users for one store selling Lady Gaga’s album: the Fame Monster

Figure 6.2: Tags as Bridge: transferring category information between different domains

We propose a unified model, which not only use social tags conveying partial and latent information about the web objects as a novel evidence to facilitate classifying
objects on the web, but also exploit the relative information among tags. The web object classification is formulated by constructing a tag relevance graph and a max-margin optimization process. Our experiments demonstrate the effectiveness of the proposed approach for web object classification.

6.2 A Discriminative Model

We formulate the web object classification problem as an optimization framework based on social tag relevance tree/graph. Rather than considering a graph of web objects and associated tags to do the web object classification [145], we focus on referring the implicit information of tags relevance graph/tree, $G = \{V, E\}$, for the task via a discriminative model. A vertex $v_i \in V$ corresponds to tag $t_i$, and an edge $(v_i, v_j) \in E$ corresponds to the dependency between each pair of tags.

6.2.1 Modeling Functions

Our model can be denoted as: $\mathcal{F}_w(x, y) = \max_t w^T f(x, t, y)$, where $f(x, t, y)$ denotes a feature vector including features of web objects, web objects’ tags $t$ and their corresponding class label $y$. The model can be decomposed as linear combination of multiple functions (see Figure 6.3) based on the energy-based learning [69]:

$$
\mathcal{F}_w(x, y) = w_1^T f_1(x) + \sum_{i \in V} w_2^T f_2(x) + \sum_{i \in V} w_3^T f_3(x) + \\
\sum_{(i, j) \in E} w_4^T f_4(t_i, t_j) + \sum_{j \in V} w_5^T f_5(t_j)
$$

The five functions are as the following.
1. Modeling Global Category-Feature Relevance: $w_1^T f_1(x)$, aims to map the relevance between category and features of web objects. It is a standard linear function for web object classification without any tags, only depending on the extracted textual features ($f_1(x)$), which represents the basic feature vector extracted from the web objects, this function is related to the base features: the broad variety of attributes requires a feature representation to describe several
possible aspects of this web object. In our work, we extract textual features from web object’s titles and use the bag of words style to represent these features.

2. Modeling Global Feature-Tag Relevance: we define it as a linear function $w^T_2 f_2(x)$, where $x_i$ is the feature vectors extracted from object $o_i$, this model is trained to predict whether web object $o_i$ has the $j$-th tag without considering the object class label and other tags.

3. Modeling Local Category-Tag Relevance: we define this model as $w^T_3 f_3(x)$ to show the tags configuration for each category label $y \in \mathcal{Y}$, and $w_3$ represents the parameters for the $i$-th tag (i.e., $t_i$) if the object category label is $y$. This local relevance reflects the dependency between each category label and its corresponding tag subsets in the tag space and can tackle the misunderstandings of semantic information of tag and its real meaning. For instance, the tag ‘Apple’ is assigned to multiple categories, laptop, fruit and company. Our model is similar with learning ‘Apple’ classifier in each category and do feature selection. For instance, learning the ‘Apple’ from the fruit which have other discriminative features like ‘eatable’. This model can uncover some specific tag combinations for each category label.

4. Modeling Pairwise Tag Relevance: it can be defined as $w^T_4 f_4(t_i, t_j)$ and which is showing the dependencies between tag $i$ and tag $j$, $f_4(t_i, t_j)$ is a sparse binary vector of length $|\mathcal{T}||\mathcal{T}|$. For example, if the $i$-th tag corresponds to Fararri and the $j$-th corresponds to Car, the $w_4$ that corresponds to $(1,1)$ and $(0,0)$ will probably tend to have large values, since it has much probability of Fararri and Car appearing together when users tag a web object.

5. Modeling Global Category-Tag Relevance: this function indicates how likely the category label being $y$ and the $j$-th tag being $t_j$. For example, if the category
is 'Music' and the $j$-th tag is 'DVD', then the function will have a larger value.

### 6.2.2 Max-Margin Learning Algorithm

Our learning method is inspired by the success of max-margin methods in machine learning [36, 69]. Given a learned model, the classification is achieved by first finding the best labeling of the web objects for each category, then picking the category label with the highest score. The learning algorithm aims to set the model parameters so that the scores of correct category labels on the training data are higher than the scores of incorrect category labels by a large margin.

Let $D = (\langle x_1, t_1, y_1 \rangle, ..., \langle x_n, t_n, y_n \rangle)$ be the training set. The goal of learning our proposed model is to learn the model parameters $w$, so that for a new testing instance (i.e., web object), we can classify it into the category $y^*$ if $y^* = \text{arg max}_y \mathcal{F}(x; y)$.

This can be achieved by solving the following optimization problem:

$$
\min \left( \frac{1}{2} \|w^T\|^2 + C \sum_{i=1}^{N} \xi_{(i)} \right) \text{s.t.} \quad w^T \mathbf{f}(x_{(i)}, t_{(i)}, y_{(i)}) - \max_{t} w^T \mathbf{f}(x_{(i)}, t, y) \\
\geq \ell(y, y_{(i)}) - \xi_{(i)}, \forall i, \forall y
$$

(6.2)

where $C$ is the trade-off parameter controlling the regularization, $\ell(y, y_{(i)})$ is a loss function indicating the cost of misclassifying $y_{(i)}$ as $y$, and $\xi_{(i)}$ is the slack variable for the $i$-th training example to handle the case of soft margin. The constraints indicates that: considering the $i$-th training, we want the score of the true category label $y_i$ and its ground-truth tags to dominate the score of any other category label $y$ over any tags.
Algorithm 5: Cutting-plane Algorithm for Learning Model Parameter $w$

**Input:** $<x_1, t_1, y_1>, ..., <x_n, t_n, y_n>$; tolerated approximation error $\epsilon \geq 0$

**Output:** $w$

1. $K = \emptyset$; $w = 0$; $\xi = 0$

2. for $i = 1$ to $n$ do

3. $K_{org} = K$

4. Optimizing $t^* = \arg\max_t w^T f(x_i, t, y)$

5. if $(w^T f(x_i, t^*, y) - w^T f(x_i, t, y)) - \ell(y, y_i) - \xi \geq 0$ then

6. $K = K \cup \{w^T f(x_i, t^*, y) - w^T f(x_i, t, y) \geq 1 - \xi\}$

7. Optimizing $\min_w \xi \left(\frac{1}{2}||w^T||^2 + C(\sum_{i=1}^N \xi(i))\right)$ s.t. $K$

8. end

9. end

10. Repeat until $K = K_{org}$

---

1. Keeping $w$ and $\xi$ fixed, optimize $t'$ for $(x_i, t, y_i)$: $t_{i, y_i} = \arg\max_{t'} w^T f(x_i, t', y_i)$.

2. Keeping $t_{i, y_i}$ fixed, solving the convex optimization problem $\min_w \xi \left(\frac{1}{2}||w^T||^2 + C(\sum_{i=1}^N \xi(i))\right)$.

In Step 1, the optimization problem can be solved efficiently for our tree-like tag structure using Viterbi dynamic programming algorithm for trees [118]. The details can be found in Section 6.2.4 For Step 2, the optimization problem is a quadratic problem that can be solved through a public package $SVM_{struct}^1$. Details can be found in Section 6.2.3.

### 6.2.3 Convex Optimization

To find the optimize parameters $w$ and $\xi$, which involves a quadratic program with piecewise linear constraints. One way to solve it is to transfer it to a standard quadratic objective function.
program with linear constraints. So Equation 6.2 can be rewritten as using cutting plane algorithm [123, 57], which starts with no constraints, then iteratively finds the most violated constraints and adds those constraints. This method can reach a solution by evaluating a polynomial number of constraints. So Equation 6.2 can be solved as follows:

1. Keeping \( w \) and \( \xi \) fixed, optimize \( t' \) for \( (x, t, y_i) \): 
   \[
   t_{i,y_i} = \arg \max_t w^T f(x_i, t, y_i).
   \]

2. Keeping \( t_{i,y_i} \) fixed, solving the convex optimization problem
   \[
   \min \left( \frac{1}{2} ||w^T||^2 + C(\sum_{i=1}^{N} \xi(i)) \right) \text{ s.t. } w^T f(x_{(i)}, t_{(i), y_i}, y_{(i)}) - w^T f(x_{(i)}, t_{i,y_i}, y) \geq \\
   \ell(y, y_{(i)}) - \xi(i), \forall i, \forall y
   \]

So step 2 consists a quadratic program with \( N \times |\mathcal{Y}| \) constraints now, which can be solved in polynomial timespan. The detailed implementation of cutting plane algorithm can be found in Algorithm 5.

6.2.4 Finding Optimal Latent Variable \( t' \)

To find optimal \( t^* \) for given object-tag pair \( (x, y) \):

\[
 t^* = \arg \max_t w \Phi(x, t, y) \tag{6.3}
\]

If the tags \( t \) forms a tree-like structure, the inference problem can be solved using Viterbi dynamic programming algorithm for trees. Then we can solve it using standard
linear programming \cite{118}. Finding the optimal $t$ can be reformulated as:

$$
\max_{0 \leq \mu \leq 1} \sum_{j \in V} \sum_{a \in H} \mu_{ja} \phi_j(a) + \sum_{jk \in \epsilon} \sum_{a \in H} \sum_{b \in H} \mu_{jkab} \phi_{jk}(a, b)
$$

subject to:

$$
\sum_{a \in H} \mu_{ja} = 1, \forall j \in V \\
\sum_{a \in H} \sum_{b \in H} \mu_{jkab} = 1, \forall jk \in \epsilon \\
\sum_{a \in H} \mu_{jkab} = \mu_{kb}, \forall jk \in \epsilon, \forall b \in H \\
\sum_{b \in H} = \mu_{ja}, \forall jk \in \epsilon, \forall a \in H
$$

(6.4)

where $\mu_{ja}$ is an indicator function $I(t_j = a)$ for all vertices $j \in V$ and their values $a \in H$. $\mu_{jkab}$ denotes $I(t_j = a, t_k = b)$ for all edges $(j, k) \in \epsilon$ and the values of their nodes, $a \in H, b \in H$. $\phi_j(t_j)$ denotes the summation of all the unary potential functions involving $j \in V$. $\phi_{jk}(t_j, t_k)$ denotes summation of all the pairwise potential functions involving edge $jk \in \epsilon$.

### 6.3 Loss Function

Training process consists in shaping the $w$, so that for any given $X$, the inference algorithm will produce the desired value for $Y$. We adopt the classic 0-1 loss function indicating the cost of misclassifying $y^{(i)}$ as $y$, and it is defined as:

$$
\ell_{0/1}(y, y^{(i)}) = \begin{cases} 
1 & \text{if } y \neq y^{(i)} \\
0 & \text{otherwise}
\end{cases}
$$

(6.5)

In the case of 0-1 loss, the cumulative loss is exactly the number of training examples incorrectly classified by the model, which is directly related to the overall training error. To be consistent with the real-world situations, where the class label distribution usually are not symmetry and shows highly skewed, we did not manually adjust the
uniform distribution in our experiment. To tackle this skewed data, we do the smoothing for the 0-1 loss function, and redefined as:

$$\Delta_r(y, y^{(i)}) = \begin{cases} \frac{1}{m_u} & \text{if } y \neq y^{(i)} \\ 0 & \text{otherwise} \end{cases}$$

(6.6)

where $m_u = \sum_v n_{uv}$, $n_{uv}$ means the number of object belong to class $u$ is assigned to class $v$. This smoothed loss function can reflect the mean accuracy of classifier.

### 6.4 Tag Tree Generation

In our current work, we treat the tag space as the structural latent variables, and explore the pairwise relative information between each pair of tags. We assume there are natural dependencies between tags pairs $(t_i, t_j)$. We summarize three relative information between tags that can be tackled by our work. Generating a tag tree needs to take the following main steps:

1. Tag correlation graph construction. We compute the normalize mutual information for the dependency between tags to measure similarity between each pair of tags, and it can be obtained as

$$NormMI(i, j) = \frac{MI(i, j)}{\min\{H(i), H(j)\}}$$

(6.7)

where $MI(i, j)$ is the mutual information of the tag pair $(i, j)$, it is defined by:

$$MI(i, j) = \sum_{y_i, y_j} Pr(y_i, y_j) \log \frac{Pr(y_i, y_j)}{Pr(y_i)Pr(y_j)}$$

(6.8)

and $H(i)$ is the entropy of tag $i$ defined by:

$$H(i) = -\sum_{y_i} Pr(y_i) \log Pr(y_i)$$

(6.9)
all the probability can be calculated from the dataset. It should be noted that we estimate these probaiblity from the whole dataset, other than only from the ground-truth of the training dataset. With normalized mutual information between tag pairs, a tag correlation graph $S$ can be formed.

2. Tag Correlation graph clustering. After obtaining the symmetric similarity graph of tags $S$ from the Step 1, then we adopt the spectral clustering algorithm with k-means to cluster the tags. The basic implementation steps in our approach are as the following:

(a) Calculating top k eigenvectors: Once we have obtained a sparse similarity matrix $S$, we can construct its normalized Laplacian matrix $L$:

$$L = I - D^{1/2}SD^{-1/2}$$

where $D$ is a diagonal matrix with $D_{ii} = \sum_{j=1}^{n} S_{ij}$, $D^{-1/2}$ is the inverse square root of $D$. To compute the eigenvectors by Lanczos/Arnoldi factorization, we adopt the efficient numerical sparse eigensolvers ARPACK [70]. Then we can construct matrix $V$ whose columns are the k eigenvectors.

(b) Computing normalized matrix $U$: $U_{ij} = \frac{V_{ij}}{\sqrt{\sum_{r=1}^{k} V_{ir}^2}}$, each row of $U$ has a unit length.

(c) Use $k$-means algorithm to cluster $n$ rows of $U$ into $k$ clusters: $U'$s $n$ rows correspond to $k$ orthogonal points on the unit sphere. So that, $n$ rows of $U$ can be easily clustered by simple clustering techniques.

3. Reference tag clustering tree construction. There are two main stages to achieve in this step.
(a) Calculating edge weight amongst clusters. We treat the tag clusters as reference nodes, for calculating the edge weight between each pair of reference nodes. We sum all the weight of edge for tags which are in each cluster together as the overall weight between each pair of clusters. Finally, we can get the mutual relations among these $k$ clusters.

(b) Constructing tag tree. We run the minimum spanning tree to generate the tag tree. A spanning tree has exactly $n - 1$ ($n$ is the total number of vertices on the tag correlation graph) edges. The spanning tree with smallest total weight (weight is the NormMI of each each pair of vertices) is called the minimum spanning tree.

6.5 Discussion: Learning with Invisible Side Information

Our framework also can handle the training instances without tags or partial tags, in this case, the way to learn parameters $w$ can be formulated as:

$$\min \left( \frac{\lambda}{2} ||w_i^T||^2 + C(\sum_{i=1}^{N} \zeta(i)) \right)$$

$$s.t. \max_t w_i^T f(x^{(i)}, t, y^{(i)}) - \max_t w_i^T f(x^{(i)}, t, y) \geq \delta(y, y^{(i)}) - \zeta(i), \forall i, \forall y$$

(6.11)

since $t$ is unobservable in this training stage, and it can be treated as a latent variables and using latent SVM formulation [36]. There are many methods of solving the non-convex optimization problem, we adopt the approach in [30] and rewrite Equation 6.11:

$$\minimize_w \mathcal{L}(w) = \frac{\lambda}{2} ||w||^2 + R(w)$$

(6.12)

where $R(w)$ is an upper bound of the empirical risk that needs to be minimized. This can be solved iteratively building an increasingly accurate piecewise quadratic approx-
6.6 Experiments

We have conducted several experiments to validate the effectiveness of our approach based on real life data. In this section, we mainly introduce the experimental environment setup and present the experimental results. All experiments were conducted in C++, Matlab and C# on a Core 2 Quad 2.70 GHz machine with 8GB RAM.

6.6.1 Data Preparation

We use the dataset in [145] (5,536 web pages from the ODP shopping ² and 6,123 product information in 8 different categories from Amazon ³. We also collected 461 video clips from YouTube. For each web object, its associated tags, description, and category information were also collected. We performed some pre-processing of the tags and then constructed the tag relevance graph.

Basically, for web pages and products collection, we use the dataset in [145] collected from Amazon ⁴ as the initial dataset. This dataset contains 6123 products information, and 5536 web pages from the ODP shopping ⁵. Furthermore, for highlighting the heterogeneous objects, since the Web has been evolving from Web 2.0 to Web 3.0, more and more web services e.g., SOAP web services, REST web services are delivered to the Web. We collected 5960 web services (APIs) from the Programmableweb ⁶ and their associated descriptions, tags, category information, and API names are also collected. According to the Amazon dataset category, we select the corresponding

²http://www.dmoz.org/Shopping/
³www.amazon.com
⁴http://www.cs.illinois.edu/homes/zyin3/
⁵http://www.amazon.com/
⁶http://www.programmableweb.com/apis/directory
categorical Web services into the mixed dataset. Then, the videos collection is from Youtube \(^7\), we collect the categories of Youtube *Pets/Animals* and *Music*, for videos from other categories, like 'Jewelry', there is no category information in Youtube category homepage, in this case, we manually collect 50 by query the keywords "Jewelry". All the videos contain the tags, and their brief description and category information.

The table shows the data statistics. The data is shown in Table 6.1:

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Webpages</th>
<th>Products</th>
<th>Webservices/APIs</th>
<th>Videos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Source</td>
<td>ODP:Shopping</td>
<td>Amazon</td>
<td>ProgrammableWeb</td>
<td>Youtube</td>
</tr>
<tr>
<td>Category</td>
<td>#</td>
<td>#</td>
<td>#</td>
<td>#</td>
</tr>
<tr>
<td>Publications/Books</td>
<td>558</td>
<td>937</td>
<td>98</td>
<td>N/A</td>
</tr>
<tr>
<td>Electronics</td>
<td>494</td>
<td>945</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Health/Medical</td>
<td>1009</td>
<td>747</td>
<td>58</td>
<td>50</td>
</tr>
<tr>
<td>Home/Garden/Estate</td>
<td>1976</td>
<td>841</td>
<td>41</td>
<td>41</td>
</tr>
<tr>
<td>Jewelry</td>
<td>452</td>
<td>386</td>
<td>N/A</td>
<td>20</td>
</tr>
<tr>
<td>Music</td>
<td>527</td>
<td>944</td>
<td>151</td>
<td>200</td>
</tr>
<tr>
<td>Office</td>
<td>77</td>
<td>695</td>
<td>66</td>
<td>N/A</td>
</tr>
<tr>
<td>Pet/Animal</td>
<td>443</td>
<td>628</td>
<td>N/A</td>
<td>150</td>
</tr>
</tbody>
</table>

### 6.6.2 Tag Preprocessing

Each web object may associate with one or multiple tags. We preprocess the tags given in the initial dataset, and build up a tag book so that we can generate a tag relevance graph for our model. We consider the tags from two perspectives: tags recommended by Amazon, which indicate the confidences and also the customers associated tags. Then, we do the preprocessing work to narrow down and refine the tags space before we build a tag tree as described in Section 6.4.

The detailed information on the process of preprocessing tags is as the following:

\(^7\)http://www.youtube.com
1. We need to do some preprocessing work and filter the trustworthy tags, and prune the vocabulary by stemming each term to its root, removing function terms (like adjective words, and only keep the meaningful nouns), and removing terms that occurred and also remove the occurrences of tags in one product is less than 5, which refers to less reputation. Next, we unite the similar meaning tags using the language processing tool *Catvar 2.1*[^1], which is a Categorial-Variation Database (or Catvar) is a database of clusters of uninflected words (lexemes) and their categorial (i.e. part-of-speech) variants. For example, the words hunger(V), hunger(N), hungry(AJ) and hungriness(N) are different English variants of some underlying concept describing the state of being hungry. Another example is the developing cluster:(develop(V), developer(N), developed(AJ), developing(N), developing(AJ), development(N))[^2]. e.g., the words have different shape but same root: baby and babies etc or *children book* and *kid book* etc.

2. Meaningless symbols are removed. Since the tags are given by end users, it is normal that there might exist some meaningless symbols or repeated words. Moreover, the products whose number of tags are less than 5 are also removed. Some words can also be merged manually. For example, *childhood memory* merge to *child*, or *best cookbook* to *cookbook*, *dog training* to *dog behavior* in the category *PET*. We extract the condensed tags as the attributes for the web objects, and then we make the connection between the condensed attributes and original tags. After complete the processing all tags, we obtain the refined tag set to construct the tag relevance graph using our approach described in Section 6.4.

To evaluate the average performance across multiple categories, the micro-averaging and macro-averaging $F_1$ are introduced. The micro-averaged scores tend to be domi-

nated by the performance on common categories, while the macro-averaged scores are influenced by the performance in rare categories.

6.6.3 Performance Comparison

In the experiments, we demonstrate that our classification model based on social tagging performs better than other classification methods. Specially, our model can solve those problems that we encounter in web object classification. We show the following. First, social tags provide an ideal feature space for web object classification compared with other features. Second, our model makes good use of the link structure of objects and tags, which effectively capture the interconnections of objects through the social tags. Third, our method is effective when there is few or no label available for web objects. Besides, prior knowledge of unlabeled objects can be incorporated, so our model can be combined with other classification methods seamlessly. Furthermore, we show that our algorithm is efficient and can converge after several rounds of iterations in different scenarios.

6.6.3.1 Overall Performance

In the experiment, we compare our methods with two baseline methods, namely i) SVM + Title, and ii) SVM + Title + Tag.

<table>
<thead>
<tr>
<th>Measure</th>
<th>1% MicroF1</th>
<th>1% MacroF1</th>
<th>5% MicroF1</th>
<th>5% MacroF1</th>
<th>10% MicroF1</th>
<th>10% MacroF1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM + Title</td>
<td>0.4475+0.73</td>
<td>0.4233+0.61</td>
<td>0.6260+0.69</td>
<td>0.6103+0.57</td>
<td>0.6764+0.36</td>
<td>0.6573+0.41</td>
</tr>
<tr>
<td>SVM + Title + Tag</td>
<td>0.4634+0.99</td>
<td>0.4347+0.82</td>
<td>0.6671+0.84</td>
<td>0.6254+0.78</td>
<td>0.6938+0.47</td>
<td>0.6747+0.38</td>
</tr>
<tr>
<td>Our Approach</td>
<td>0.7735+0.33</td>
<td>0.7629+0.27</td>
<td>0.7905+0.40</td>
<td>0.7737+0.33</td>
<td>0.8028+0.23</td>
<td>0.7906+0.21</td>
</tr>
</tbody>
</table>

We take 1%, 5% and 10% of the dataset as the training set, and the rest for test-
ing. Table 6.2 shows the results. From the table we can see that, for the SVM-based methods, using tags and titles achieves better results than only using titles. Our model outperforms both SVM-based methods. Also, the performance of SVM-based methods are more sensitive to the label ratio comparing with our model.

### 6.6.3.2 Robustness Evaluation

The second experiment studies the robustness of our model when the base features are weak. In the default base feature function settings, we use the textual information extracted from the product web page including title, description and product details. Unfortunately, many web objects may not have meaningful descriptive information. From Figure 6.4 we can see that there is no much fluctuation in performance when partial (i.e., only title) or full textual features (i.e., title and description) are used. This indicates our tag-centric model is robust and not sensitive to features.

![Figure 6.4: Our Tag-Centric Model with Different Base Feature Settings: Partial Textual Features (Title Only) vs. Full Textual Feature (Title+Description)](image-url)
6.7 Related Work

In this section, we review some research work related to the approach proposed in this chapter. Our work is related to the web object classification, which can be boiled down as inferring the unlabeled objects based on labeled objects. Many work explore the association between the social tags and web pages, for example [99] exploits the correlations between tags and their topics based on latent topic model.

Given only a small portion of labeled data and a large amount of unlabeled data, unlabeled data can be combined with the labeled data together to improve the accuracy of the classifier. Zhu et al. [161] formulated the learning problem as a Gaussian random field on a weighted graph of labeled and unlabeled objects. Zhou et al. [157] proposed a semi-supervised learning framework to achieve local and global consistency. Both methods model the classification problem on a graph with objects as vertices, our work is similar with [145], we also consider the tags form a tag graph, but in [145], they did not explore the relevance among labels, and our focus is exploring the relevance among different tags.

Our work also is related to the collective multilabel classification on exploring local dependency of labels. Like [42] using conditional random fields (CRF), [43] proposes a correlated multi-label feature selection algorithm for learning the label correlation and selecting the features simultaneously. However, collective inference and semi-supervised learning on graphs are limited on capturing the local dependency of nodes in the relational network. Some work focus on considering the dependency between labels [78] while many other researches work on capturing the long-distance relevance of nodes. For example, in [133], Xu et al. propose a nonparametric infinite hidden relational model to capture the autocorrelation. In [88], Neville and Jensen use clustering algorithm to find cluster membership and fix the latent group variables for inference.
However, both approaches are based on Markov assumptions and not suitable for large networks due to their complexity and high computational cost for inference. [112] proposed an algorithm for collective inferencing by exploring latent linkages among the objects in the form of a latent graph.

Our work is also related to the social tagging and tags. The authors in [10] explore the social tagging on optimizing web search. [17] explores the social annotations using tags on web page clustering. Wu et al. [131] use a probabilistic model to extract semantic concept by exploring the links between tags and pages. Our work is closely related to the work in [145], which formulates the problem of web object classification as an optimization problem based on a social tagging graph. We do not consider the graph structure of social tags. Instead we model web objects as the basic textual features and a set of tags, which are then formulated using the max-margin formalism in the discriminative learning.

### 6.8 Conclusion

In this chapter, we have presented a discriminative model for classifying web objects by modeling the objects category labels and their corresponding tags. We formulate this model into a max-margin framework with structured latent variables. Our model encapsulates the relevance among the social tags via the tag relevance graph. We formulate our model into the max-margin learning framework, and learned based on cutting plane method. We validate our approach using a dataset that contains heterogeneous real-life web objects, e.g., products, web pages, images, and videos. Our experimental results show the good performance of our model.
Chapter 7

Particle Filtering based Services

Availability Estimation

Services provided by physical things have one unique inherent feature, which may be more highly dynamic comparing to traditional services since the resources and services exposed in Internet of Things can be disconnected and vanished at any time and become more dynamic since user can restart or stop their working as the corresponding devices come and go. In this chapter, we study the unique attribute of things and propose a particle-filtering based algorithm to predict and track the things availability in real-time, which significantly improves the selection of appropriate things for specific tasks.

Our contributions are summarized as follows:

- We propose a model for assessing services availability using particle filter technique, which can return precise and dynamic prediction of this availability. In our approach, the availability of services offered by things combines both historical availability information and predicted availability.

- We design an algorithm to optimize service selection from a cluster of services (like things community where all things offer the services which have similar functionalities) by dynamically reducing the candidate services search space.
during things composition. Top services with high availability can be always maintained for each thing community, which are recommended to composite services, thereby not only ensuring the high availability of composite things for value-added services, but also significantly improving the efficiency of things composition.

- An implementation of a research prototype system using a number of state-of-the-art technologies. To validate the feasibility and benefits of our approach, we conducted extensive experimental studies.

### 7.1 Motivations and Challenges

Services offered by things in Internet of Things era are highly dynamic and their availability might changes over time. Guaranteeing the availability of a service is a significant challenge due to the varying number of invocation requests the service has to handle at a time, as well as the dynamic nature of the Web and physical things. Over the last few years, many works have emerged in addressing services availability problem. Almost all of these approaches are based on the concept of service community which is a community of things providing similar functionalities (but different non-functional properties such as different availability) [87, 13, 148] are grouped in a particular “cluster”. The basic idea on improving the availability of service in a composition is to substitute services with poor quality using peers with better quality from the same service community.

Most service selection approaches assume that the availability of service is pre-existing and readily accessible. This unfortunately is not the case in most real world applications. In reality, the availability status, as well as other non-functional properties, of a service is highly uncertain, which changes over time. How to accurately estimate
and predict the availability status of a service becomes an important research problem. In addition, given advocating adoption and development of Internet of Things, more and more services offered by things will be available and the size of service communities will be inevitable large. Selecting from such a large space will inevitably lead into performance problems. Ideally, low quality services should be automatically filtered and not be considered during service composition.

### 7.2 Model Overview

We propose a framework which can adaptively maintain a dynamic filtered sets of services. Our proposed work can continuously monitor and furtherer estimate the state of availability of services and dynamically generate and maintain the selected services subsets, these subsets belong to each services classifications and ranked according to their availability. Under this framework, services set containing high availability services can be guaranteed to maintain in real-time, which can serve more effective things composition for value-added services and also optimize services selection from a cluster of services (like service community where all things offer similar functionalities) by dynamically reducing the candidate services search space during services composition. Top services with high availability can be always maintained for each service community, which are recommended to composite services, thereby not only ensuring the high availability of composite services, but also significantly improving the efficiency of service composition.

Figure 7.1 shows the basic idea of our approach. Specifically, we propose to add a filtering layer between service supplier layer and higher layer, i.e., composition layer (right side of Figure 7.1). The layer of services contains several service clusters and each of them consists of services with similar functionalities.
The filtering layer is essentially a subset of service clusters in the service layer, which consists of services with high availability that will directly involve in higher service operations. Our approach dynamically adjusts the members in the filtered service clusters where degrading services are automatically and transparently replaced with services of better availability from the service layer.

In the following sections, we introduce the particle-based filtering algorithm of services availability.

### 7.3 Availability Modeling

There are different classifications of availability and many ways to calculate it [35]. Almost all existing approaches (e.g., [149, 79, 45]) use operational availability that measures the average availability over a period of time (i.e., the ratio of the service
uptime to total time). Although this is simple to calculate, it is hard to measure the availability of a service at a specific time.

In this work, we model service availability as *instantaneous* (or point) availability. The instantaneous availability of a service $s$ is the probability that $s$ will be operational (i.e., up and running) at a specific time $t$. The following discusses how this is calculated.

At a given time $t$, a service $s$ will be available if it satisfies one of the following conditions:

- The service $s$ is working in the time frame of $[0, t]$ (i.e., it never fails by time $t$). We represent the probability of this case as $\mathcal{R}(s, t) = \frac{T_a(t)}{T_s(t)}$, where $T_a(t)$ is the total available time for each component service and $T_s(t)$ is the total measurement time $[100]$.

- The service $s$ works properly since the latest repair that occurred at time $u$ ($0 < u < t$). The probability of this condition is $\int_0^t \mathcal{R}(s, t - u)m(s, u)du$, where $m(s, u)$ is the renewal density function of $s$. In our work, we model $m(s, u)$ as Poisson distribution $m(s, u) \sim \frac{e^{-\lambda} \lambda^k}{k!}$.

Based on these two conditions, the availability of $s$ at time $t$, $A(s, t)$, is calculated using the following formula:

$$A(s, t) = \mathcal{R}(s, t) + \int_0^t \mathcal{R}(s, t - u)m(s, u)du$$  \hspace{1cm} (7.1)
7.4 The Particle Filter Based Approach

7.4.1 Particle Filtering

We consider the availability of services as a dynamic system (i.e., it changes over time), which can be modeled as two equations: state transition equation and measurement equation. The states can not be observed directly and need to be estimated, while the measurements can be observed directly. For a very simple example, if we track a robot, we can model its state as a vector including the robot’s position and velocity \( \{p, v\} \), and the observation of the position (i.e., measurements) can be obtained from the GPS. For services, [45] exploits Extended Kalman Filter to predict the service dependability state. In our work, we model the component service availability state as:

\[
x_t = f_t(x_{t-1}, v_{t-1})
\]

(7.2)

where \( f_t \) is a non-linear state transition function of the availability of a component service, \( x_t, x_{t-1} \) are estimated and previous states of the component services respectively, and \( v_{t-1} \) is the state noise in a non-Gaussian distribution (e.g., disturbance caused by network throughput to the service availability). Similarly, measurement equation for the component service availability is represented as

\[
z_t = h_t(x_t, n_t)
\]

(7.3)

where \( h_t \) is a non-linear measurement function, \( z_t \) is a measurement, \( x_t \) is the estimated service availability state, and \( n_t \) is the measurement noise which is not confined as Gaussian distribution, (e.g., observation error).

The availability of services changes over time, which is full of uncertainty due to problems of network issues, hosting servers’ loads, and even service requester environments. However, the state transition of availability from time \( t - 1 \) to time \( t \) can
not be guaranteed as a linear transition, and in the measurement equation, the noise can also not be guaranteed as Gaussian distribution. We therefore propose to exploit the generic particle filter [62] to solve the dynamic availability of services. Particle filtering can deal with the non-linear and non-gaussian distribution situation presented in services, which will be detailed later.

The reasons backing particle filter adoption are as follows:

- Particle filters can represent arbitrary probability densities by a collection of
particles with weight;

- Unlike Kalman filters, particle filters can converge to the true posterior even in non-Gaussian, non-linear dynamic systems; and

- Compared to grid-based approaches, particle filters are very efficient because they automatically focus their resources (particles) on regions in state space with high probability.

Briefly, the particle filter is a technique for implementing Bayesian filter recursively by Monte Carlo sampling, and it is a sequential Monte Carlo methods based on particles representations of probability densities other than the Gaussian distribution which can be used in more general areas and for any state space model [6, 91]. The particle filter aims at tracking the state of a system as it evolves over time and typically with a non-Gaussian and potentially multi-model probability density function (pdf). It represents the pdf as particles which are associated with weight, and estimates the states by recursively updating approximations of posterior. Figure 7.2 shows the basic implementation process of the generic particle filter, consisting the following three main processes:

1. **Particle generation:** draw $N$ particles with weights for state from a proposal distribution function, the proposal distribution function can be defined freely (e.g., uniform distribution).

2. **Weight update:** the weights of particles are recursively updated and normalized.

3. **Resampling:** when implementing the generic particle filter, after a few iterations, most of particles have negligible weight. In other words, the weight is only concentrated on a few particles. The resampling process stochastically discards
Chapter 7. Particle Filtering based Services Availability Estimation

**Algorithm 6: Generic Particle Filter Algorithm**

1. Let $N_{eff}$ be the effective particle sample size and $N_t$ be the threshold of the particle size.
2. for $i = 1 : N_s$ do
   3. Draw $x_i^t \sim q(x_i^t | x_{i-1}^t, z_t)$
   4. Assign the particle a weight, $w_i^t$ according to $w_i^t \propto \frac{p(x_i^t | z_t) p(x_{i-1}^t | z_{i-1})}{q(x_i^t | x_{i-1}^t, z_t)}$, $q(\cdot)$ is a proposal function and can be defined.
5. end
6. Calculate total weight: $t = \sum_{i=1}^{N_s} w_i^t$
7. for $i = 1 : N_s$ do
   8. Normalize: $w_i^t = \frac{w_i^t}{t}$
9. end
10. Calculate $N_{eff}$ using $N_{eff} = \frac{1}{\sum_{i=1}^{N_s} (w_i^t)^2}$
11. if $N_{eff} < N_t$ then
12. Resample (Algorithm 7).
13. end

**Algorithm 7: Resampling Algorithm**

1. Let CDF be the Cumulative Density Function.
2. Initialize CDF: $c_1 = 0$;
3. for $i = 2 : N_s$ do
   4. Construct CDF: $c_i = c_{i-1} + w_i^t$;
5. end
6. Start at the bottom of the CDF: $i = 1$
7. Draw a starting point: $u_1 \sim U[0, N_s^{-1}]$
8. for $j = 1 : N_s$ do
   9. Move along the CDF: $u_j = u_1 + N_s^{-1} (j - 1)$
   10. while $u_j > c_i$ do
       11. $i = i + 1$;
    12. end
   13. Assign sample: $x_i^t = x_i^t$;
14. Assign weight: $w_i^t = N_s^{-1}$;
15. Assign parent: $i' = i$;
16. end

particles with negligible weight, and replaces them with the particles with large weights.

Algorithm 6 shows the detailed algorithm of the generic particle filter. The re-
sampling algorithm, which is also called *systematic resampling* and is simple to implement, is shown in Algorithm 7. Its time complexity is $O(N_s)$ where $\mathcal{U}[a, b]$ is the uniform distribution on the interval $[a, b]$. Interested readers are referred to [62] for more details.

### 7.4.2 Particle Filtering Based Estimation

Services’ availability state is highly dynamic and therefore needs an adaptive approach to monitor and track each service’s state. This is important to conduct optimized selection algorithms for composite services such as [148, 4]. We apply the particle filtering technique to make accurate estimation of service’s availability state, which serves the foundation for dynamically optimized selection of things in services composition.

We consider that the changes of availability of services are uncertain. The availability modeling function is non-linear and the noise (Section 7.4.1) cannot be guaranteed as a Gaussian distribution. Particle filter can improve the performance over the established non-linear filtering approaches since it provides optimal estimation in non-linear and non-Gaussian state space models, as well as estimation of non-linear models without making any assumption on the measurement noise distribution. Particle filter can estimate a system states sufficiently when the number of particles (estimations of the state vectors that evolve in parallel) is large.

The particle filter refers to belief using a number of particles. There are two main steps in the particle filter algorithm: *prediction* and *update*. Particle filters realize Bayes filter updates according to a sampling procedure, often called *sequential importance sampling with resampling* [40]. Whenever new observations $z_t$ are discovered, the filter predicts the state using $Bel^- \leftarrow \int p(x_t|x_{t-1}) Bel(x_{t-1}) dx_{t-1}$. And then the filter will correct the predicted estimation using $Bel(x_t) \leftarrow \alpha_t p(z_t|x_t) Bel^-(x_t)$, where $Bel(x_t)$ is a probability distribution over $x_t$. 
In our approach, we model the availability of a service \( i \) at time \( t \) as \( x_i(t) \), which maintains the probability distribution of the service availability at \( t \). The state transition of service \( i \)'s availability can be represented as:

\[
x_i(t + 1) = g(x_i(t)) + \phi_i(t)
\]

where \( g(x_i(t)) \) denotes the nonlinear transition of service \( i \)'s availability and \( \phi_i(t) \) denotes the noise to service \( i \)'s availability. We can further define the observation equation of the service \( i \)'s availability as:

\[
z_i(t) = h(x_i(t)) + \delta_i(t)
\]

where \( z_i(t) \) is the observation value of service \( i \)'s availability, \( h(x_i(t)) \) is the observation function, and \( \delta_i(t) \) is the observation noise. In our particle filtering approach, the posterior distribution of \( x_i(t) \) can be inducted as the belief \( Bel(x_i(t)) = \{x_i(t), w_i(t)\} \), \( i = 1, 2, ..., M \), where \( w_i(t) \) are the different weight values, which indicate the contribution of the particle to the overall estimation, also called important factors \( \sum w_i(t) = 1 \).

Algorithm 8 shows steps to summarize the particle filtering process. Firstly, we initialize a uniformly distributed sample set representing a service’s availability state. We assign each sample a same weight \( w \). Secondly, when the availability changes, the particle filter calculates the measurement by adjusting and normalizing each sample’s weight. These samples’ weights are proportional to the observation likelihood \( p(z|x) \). The particle filters randomly draw samples from the current sample set whose probability can be given by the weights. Then we can apply the particle filters to estimate the possible next availability state for each new particle. The prediction and update steps will keep running until convergence.

We calculate the weight distribution by considering the bias resulted from the routing information between users and targeting component services (e.g., routing-hops
Chapter 7. Particle Filtering based Services Availability Estimation

Algorithm 8: Particle Filter based Algorithm

1. Initialization: compute the weight distribution $D_w(a)$ according to IP address distribution.

2. Generation: generate the particle set and assign the particle set weight, which means $N$ discrete hypothesis.
   - generate initial particle set $P_0$ which has $N$ particles, $P_0 = (p_{0,0}, p_{0,1}, \ldots, p_{0,N-1})$ and distribute them in a uniform distribution in the initial stage. Particle $p_{0,k} = (a_{0,k}, weight_{0,k})$ where $a$ represents the service availability.
   - assign weight to the particles according to our weight distribution $D_w(a)$.

3. Resampling:
   - Resample $N$ particles from the particle set from a particle set $P_t$ using weights of each particles, refer to Algorithm 7.
   - generate new particle set $P_{t+1}$ and assign weight according to $D_w(a)$.

4. Estimation: predict new availability of the particle set $P_t$ based on availability function $f(t)$.

5. Update:
   - recalculate the weight of $P_t$ based on measurement $m_a$, $w_{t,k} = \prod(D_w(a_{t,k}))(\frac{1}{\sqrt{2\pi}\phi})exp(-\frac{\delta a_{t,k}^2}{2\phi^2})$, where $\delta a_{t,k} = m_a - a_{t,k}$
   - calculate current availability by mean value of $p_t(a_t)$.

6. Go to step 3 and iteration until convergence.

between the user and the component service or whether user and targeting services are in the same IP address segment). The Sequential Importance Sampling (SIS) algorithm is a Monte Carlo method that forms the basis for particle filters. The SIS algorithm consists of recursive propagation of the weights and support points as each measurement is received sequentially. To tackle the degeneracy problem, we adopt a more advanced algorithm with resampling [6]. It has less time complexity and minimizes the Monte-Carlo variation. The resampling algorithm is given in Algorithm 7.

7.4.3 The Dynamic Tracking Algorithm

Based on the algorithm mentioned above (i.e., Algorithm 8), we can sort the top $k$ services with high availability according to the monitoring and prediction. We call
Algorithm 9: Overall Adaptive Filtering Algorithm

1. **Input**: initial availability values, \( \alpha, \tau \).
2. **Output**: predicted availability, referencing availability, candidate list.
3. 1. Read in the initial parameters;
4. 2. Calculate each values for service \( a_{ij}(s, t) \) in Web service community \( j \) at time \( t \);
5. 3. Predict the availability state of next time slot using particle filter (Algorithm 8);
6. 4. Looking up database and calculate the mean values of availability \( \mathcal{H} \).
7. 5. Calculating the reference availability \( \mathcal{R} \).
8. 6. Update the top \( k \) candidate list in each services community for every time interval \( \tau \);
9. 7. Go to step 2, and iterating.

This estimated availability \( \mathcal{E}_i \). In addition, for the overall filtering algorithm, we also take the history information on availability \( \mathcal{H}_i \) into account, on top of the estimated availability by using the particle filter technique. The historical fluctuation of services availability has important impact on the current availability of the services. We call this historical fluctuation \( \mathcal{H} \) impact as **availability reputation**. The most common and effective numerical measure of the center tendency is using the **mean**, however, it is sensitive to the extreme values (e.g., outliers) \[48\]. In our work, we define the final availability of a service as **reference availability** \( \mathcal{R} \), which is calculated using:

\[
\mathcal{R}_i(\tau) = \sigma \mathcal{E}_i(\tau) + (1 - \sigma) \mathcal{H}_i \left( \sum_{1}^{\tau-1} (\tau - 1) \right)
\]

(7.6)

where \( \tau \) is a time span which can be defined by users and \( \sigma \in [0, 1] \) is the weight and users can assign different weight based on their different preference between prediction and history of service availability. For example, if \( \sigma \) is 1, the availability of a service will totally depend on the estimation value obtained by the particle filtering algorithm. Here, the historical values can be considered as the smoother for the reference availability \( \mathcal{R} \). Finally, we summarize the overall particle filter algorithm for service selection in Algorithm 9.
7.5 Experiments

In this section, we report on some experimental results of our proposed approach. We conduct experiments not only on accuracy study of our approach, but also test our approach in composition scenario. The former aims at studying the performance of our approach, the latter aims at demonstrating an application scenario of our approach and our experimental results show the feasibility and effectiveness of our approach.

The first one studies the estimation accuracy of our approach. The second experiment studies the impact of $\sigma$ on the error rate of estimation accuracy. The third experiment compares the availability of composite services with and without our approach. The last experiment studies the performance in things composition for value-added services using our particle filter based approach.

For the experiments, we simulated 500 services of five different service communities (i.e., 100 services for each service community). We set the failure probability for the services as 3.5 percent, which complies with the findings in [60].

*Estimation Accuracy.* The purpose of this experiment is to study the accuracy of our availability estimation approach. In the experiment, we simulated services’ availability fluctuation and tracked their fluctuation of availability for 50 time steps (each time step counted as an *epoch*). The actual availability of services and corresponding estimated availability using our particle filter approach were collected and compared. Figure 7.3 shows the result of one particular service. From the figure, we can see that our approach works well in tracing and predicting the availability of services.

*Availability of Composite Services.* The purpose of the second experiment is to study the impact of our approach on the availability of composite services. We randomly generated composite services by composing services from five different communities.
We simulated a comparatively significant fluctuation on the availability (i.e., changes in availability) of services for 50 different rounds and collected the availability information of the composite services under the situations of i) using our approach and ii) without using our approach. In our experiment, the availability of a composite service, $A_c$, is a product of $e^{A(s_i, t)}$, where $c$ is a composite service, $s_i$ is a component service of $c$, and $A(s_i, t)$ is the availability of component service $s_i$.

Figure 7.4 shows the availability of a particular composite service. From the figure we can see that the availability of the composite service is more stable when using our approach. In contrast, without using our approach, its availability is very sensitive to the fluctuations of service availability. The reason is that our particle filter based approach can dynamically predict the availability of component services and proactively substitute the services with poor availability.
Chapter 7. Particle Filtering based Services Availability Estimation

Figure 7.4: Availability of a composite service

The impact of $\sigma$ for average error rate on accuracy. This experiment aims to study the impact of $\sigma$ (Equation 7.6) on the accuracy of availability estimation. In the equation, the value of $\sigma$ represents the weight between the predicted availability and historical availability. In particular, the weight of the historical availability (i.e., 1-sigma) is considered to be a smoother. In this experiment, we set the smoother over a range of 0 and 0.8 to show the impact on the accuracy of prediction for services. Figure 7.5 shows the result of a particular service. From Figure 7.5 we can see that although the error rate stays relatively stable when the smoother is less than 0.2, the average availability error rate increases constantly when the smoother becomes bigger. The reason is that the role of historical data played in the particle filtering prediction process, which is based on the Markov assumption.

We also studied the impact of $\sigma$ on the availability of composite services. In the
experiment, we set the values of the smoother as 0, 0.2, and 0.5. Figure 7.6 shows the result of one composite service. From the figure we can see that when we take the smoother into account, the availability of the composite service is more stable. Interestingly, there are no significant changes when the value of the smoother is set as 0.2 and 0.5. As a result, by combining our findings on estimation accuracy of component services (Figure 7.5), in our implementation, we chose to set the value of the smoother as 0.2.

*Time for composing services.* This experiment aims at studying the performance of our approach in services composition. In this experiment, we set the top-$N$ services where $N$ is 20, which means the size of candidate list for each service cluster is 20. We further set the number of component services to 125, 250, 375, and 500 (i.e., each service cluster has 25, 50, 75, and 100 services respectively). We recorded and compared the time used for composing composite services with and without our proposed filtering
algorithm. The availability of the composite services is manually set in this experiment ($\geq 0.80$). It should be noted that in real situation, the requirement of a composite service’s availability is usually determined by the SLA. All composite services produced similar results and Figure 7.7 shows the result of a certain composite service. It can be noticed that the improvement in reducing the execution time is obvious, particularly when the size of service communities becomes bigger. This is due to the smaller size of the filtered service communities with high quality component services.

### 7.6 Related Work

There is a large body of research work related to the topic we discussed in this paper. One important area on achieving high availability of services focuses on replication technology [105, 108, 110]. The underlying idea is to spread service replicas over var-
ious locations and if needed, direct invocation requests to appropriate replica (e.g., with lower workload). Serrano et al. [108] discuss an autonomic replication approach focusing on performance and consistency of services. Salas et al. [105] propose a replication framework for highly available services. Sheng et al. [110] further developed the idea by proposing an on-demand replication decision model that offers the solution to decide how many replicas should be created, and when and where they should be deployed in the dynamic Internet environment. While these approaches focus on improving service availability through replication, our work concentrates on monitoring and predicting service availability. Our work is complementary to these works in the sense that the estimations provide a good source of information for replication decisions.

The work presented in [45, 114, 101, 54] are the most similar ones to our work. In [45], Guo et al. model a composition process into the Markov Decision Process and
use Kalman Filter to tracking the state of composite services. Sirin et al. [114] propose a filtering methodology that exploit matchmaking algorithms to help users filter and select services based on semantic services in composition process. Rosario et al. [101] focus on Service Level Agreements (SLAs) of composite services and propose a soft probabilistic contracts that consist of a probability distribution for the considered QoS parameter. These soft contracts can be composed to yield a global probabilistic contract for composite services. However, these works focus on adaptive composition of services and do not pay attention to the availability of component services. Finally, Hwang et al. [54] propose a probability-based QoS model for describing QoS values of both atomic and composite services. Our approach uses particle filter to precisely predict the availability of services in real time and dynamically maintains a subset of services with higher availability, from which service developers can choose in their compositions.

7.7 Conclusion

Guaranteeing the availability of services offered by physical things is a significant challenge because of their dynamic natures, especially in Internet of Things era. Many existing approaches ignore the uncertain nature of service availability and simply assume that the availability information of a service is readily accessed. In this chapter, we proposed a novel approach to monitor and predict service’s availability based on particle filter techniques. Furthermore, we developed algorithms to filter services from service clusters for efficient service selection. The implementation and experimental results validated our approach. Our proposed approach in dynamic tracking things availability can be adopted in many important application scenarios in Internet of Things, e.g., effective things recommendation, which is a high level relationship between people and things. In next part, we will concentrate on things recommendation.
III

Analysis on Things Recommendation Relationship
Chapter 8

Exploring Things Recommendation in Internet of Things

With many things connected and interacting over the Internet, there is an urgent need to provide effective mechanism for things recommendation across the diverse set of things to reveal interesting patterns among them and with consideration of complex relationships (e.g., people social relationships, things correlations etc.) in Internet of Things. Efficient things recommendation can benefit many applications such as e-commerce and health care.

In this chapter, we propose a unified probabilistic factor based framework by fusing information across relationships between users (i.e., users’ social network) and things (i.e., things correlations (thanks to our previous fundamental work in Part I)) to make more accurate and effective recommendation. The proposed approach not only inherits the advantages of the matrix factorization, but also exploits the merits of user-user relations and thing-thing correlations. Our contribution can be summarized as the following:

- We propose the things recommendation framework by exploring our prior work on correlation discovery of ubiquitous things in IoT [140, 138] (Part I). The solution of this pioneering effort can contribute to multiple fields in the emerging
Chapter 8. Exploring Things Recommendation in Internet of Things

IoT era.

- We propose a unified model based on probabilistic matrix factorization, by fusing users’ social networks and things correlations simultaneously. We connect these two graphs by shared latent factors and the information in users’ relations. Things’ correlations can be uncovered and propagated to serve user-thing usage pattern prediction.

8.1 Motivations and Challenges

Recent advances in identification technologies such as radio frequency identification (RFID), wireless sensor networks, and nanotechnology make computing power available in smaller and smaller physical things. Indeed, our world is slowly evolving into an environment where everyday things such as buildings, commodities are readable, recognizable, addressable, and controllable through the Internet [111]. While it is well understood that such a ubiquitous environment offers the capability of integrating both the physical world and the virtual one, which creates tremendous business opportunities such as efficient supply chains, it also presents significant challenges. With many things connected and interacting over the Internet, there is an urgent need to provide effective mechanism for search, recommendation, annotation and classification across the diverse set of things to reveal interesting patterns among them.

Things recommendation is a crucial step for promoting and taking full advantage of Internet of Things, where it benefits the individuals, businesses and society on a daily basis in terms of two main aspects. On the one hand, it can deliver relevant things (things have similar functionalities or users might need) to users based on users’ preference and interest. On the other hand, it can also serve to optimize the time and cost of using IoT in a particular situation. Physical things in reality have multiple
unique attributes. For example, they have states (e.g., in use or not in use; expired or not expired). When a certain thing is in use, it can not be used simultaneously by another user. Under this circumstance, a recommender system can refer the user to a list of things which have same or similar functionalities. For example, if microwave 1 is in use, microwave 2 will be recommended to a user who would like to warm her food, as illustrated in Figure 8.1.

We argue that both users’ relations and things correlations should be taken into account to make accurate recommendation. Users’ relations (i.e., friendships) can have significant impact on things usage patterns. Research in [59] shows that friendships and relations between users play a substantial role in human decision making in social networks. For instance, people usually turn to a friend’s advices about a commodity (i.e., hair straighter) or a restaurant before they go for them. Sometimes, these influence from the friend circle is even more substantial than high ratings given by other people [113]. As mentioned above, things are functionality-oriented and things that have similar or same functionalities hold strong relationships. We argue that physical things have more distinctive structures and connections in terms of their functionalities in real life (i.e., usefulness), as well as non-functionalities (i.e., availability). For example, different things provide different functionalities (e.g., microwave and printer), and will be of interest to different groups of people. Pairwise things with strong correlations indicate either they have similar functionalities (i.e., microwave and roaster) or they have more likelihood to be used together. For instance, a water tap and a chop board are both in use when we prepare meals, since most of the time we need to wash cooking materials (e.g., vegetables) before chopping them. In either case, similar things usage pattern will be reflected.

To recommend things, a straightforward method is to utilize a series of discrete characteristics of things in order to recommend additional items with similar proper-
ties. However, this approach is not feasible in IoT since it is not easy to directly craft things’ profiles because things cannot be easily represented in a meaningful feature space. They usually only have very short textual descriptions and lack a uniform way of describing the properties and the services they offer [61]. Besides, things are diverse and heterogeneous in terms of functionalities, access methods or descriptions. Some things have meaningful descriptions while many others do not [25]. Utilizing a series of discrete characteristics does not take into account the influence of users’ social networks and mutual correlations of things.

Collaborative Filtering (CF) recommendation has been widely used by exploiting past behaviors between users and items (i.e., rating scores on items, click-through data or browsing history etc.) without profiling users or items [106, 74, 63, 104]. Several recent approaches based on CF also take into account the users’ social networks [81, 134] to achieve more accurate performance. However, the influence of mutual correlations between items has not been explored much yet.

8.2 Problem Description

Things recommendation in IoT can be formulated as predicting the dyadic relationships between people and things. Our model merges three heterogeneous types of dyadic relationships, namely people to things ($\exists y_{i,j}, \forall i \in P, j \in T$), people to people ($\exists s_{i,i'}, \forall i, i' \in P$), and things to things ($\exists t_{j,j'}, \forall j, j' \in T$), where $P$ and $T$ denote people and things respectively. Figure 8.1 shows the graphical representation of the three relationships.

We aim at recommending certain things to users according to predicted things usage value, which is reflected the usage frequency of things. In particular, a pair of people-thing instance $(i, j), \forall i \in T, \forall j \in P$ generates the interactive relationship
Figure 8.1: The graph induced from users’ social networks and things correlations. The connections consist of three types of links, including users’ friendships within users’ social network, things correlations with correlation graph of things and things usage links within user-thing interactions. Our task is to predict the dyadic relationship between users and things, in other words, recommending certain things to users.

which can be abstracted as:

\[
\{(i, j) \rightarrow y_{ij}\} \text{ where } i \in \mathcal{T}, j \in \mathcal{P}
\]

forms a matrix \( Y \in \mathbb{Y}^{\mid \mathcal{T} \mid \times \mid \mathcal{P} \mid} \). Our goal is to predict the missing entries \( y_{ij} \) given testing pair \((\tilde{i}, \tilde{j})\).

### 8.3 The Model

In this section, we introduce our unified model, where the information from social relations and things correlations are coupled simultaneously through shared latent factors learned from three matrices: the user-user friendship matrix, the thing-thing correlation matrix, and the user-thing usage matrix.
As shown in Figure 8.2, our model fuses social network, things correlations and user-thing interactions together, and incorporates three relationships: user-user connections $s_{i'i'}$, thing-thing correlations $t_{jj'}$ and user-thing interactions (thing usage) $y_{ij}$ with shared factors $u_i$ and $v_j$. We describe how to encode these three relationship matrices in our model from Section 8.3.1 to Section 8.3.3.

8.3.1 Encoding Users Friendship

We construct a directed weighted graph $G_u = (V_u, E_u)$, whose vertex set $V_u$ corresponds to users set $\{u_1, \ldots, u_m\}$, edges set $E_u$ represents the friendships between users and the range of their associated weight are in $[0, 1]$, bigger weights represent stronger ties between users. The weight $W_u$ indicates the user similarity influenced by the social links between users, reflecting the homophily (i.e., similar users may have similar interests). We use the cosine similarity to calculated $s_{i'i'}$ as follows:

$$s_{i'i'} = \frac{e^{\cos(b(i), b(i'))}}{\sum_{k \in \Omega(i)} e^{\cos(b(i), b(k))}}$$

(8.2)

where $\cos(b(i), b(i')) = \frac{b(i) \cdot b(i')}{||b(i)|| ||b(i')||}$, $\Omega(i)$ is the set of the user $i$’s friends (i.e., $j \in \Omega(i)$), $b(i)$ is the binary vector of things used by user $i$, $|| \cdot ||$ is the L-2 norm of vector $b(\cdot)$, and $\alpha$ is a parameter that reflects the preference for transitioning to a user
who interacts with the same things.

After we obtain the users friendship matrix from $G_u$, we factorize users’ friendship matrix to derive a high-quality, low dimensional feature representation to user-based latent features vectors $u_i \in \mathbb{R}^{1 \times m}$ and factor-based latent feature vectors $u_i' \in \mathbb{R}^{1 \times m}$ on analyzing the social network graph $G_u$. The conditional probability of $s_{ii'}$ over the observed social network is determined by:

$$s_{ii'} \sim Pr(s_{ii'}|u_i^T u_i'; \sigma_s)$$

(8.3)

where $u_i \sim N(0, \sigma_u^2)$, $u_i' \sim N(0, \sigma_u'^2)$

Similar to the Web link adjacency, if a user $i$ has lots of links to other users, the trust value of $s_{ii'}$ should be decreased. While if a user $i$ is trusted by many other users, the trust value $s_{ii'}$ should be increased, since the user can be considered as local authority, so $s_{ii'}$ should be adjusted as:

$$s_{ii'}^* = \sqrt{\frac{d^-(i')}{d^+(i) + d^-(i')}} \times S_{ii'}$$

(8.4)

where $d^+(i)$ represents the outdegree of node $i$, while $d^-(i')$ indicates the indegree of $i'$. Equation 8.3 can be reformulated as:

$$s_{ii'}^* \sim Pr(s_{ii'}^*|u^T u'; \sigma_s)$$

(8.5)

### 8.3.2 Encoding Things Correlations

Things correlation value $R_{o_i, o_i}$ are not handy to obtain, and there are some unique challenges in order to learn things correlations. To tackle this problem, we have developed a graph-based model to predict correlations [140], which is briefly described here. We derive correlations among things by mining the history of things usage events. In
Chapter 8. Exploring Things Recommendation in Internet of Things

In particular, we construct a spatio-temporal graph \( G_m \) and a social graph \( G_u \) to model things usage contextual information and interactive relationships between users and things. The spatio-temporal graph captures the spatial and temporal information in things usage events, i.e., where and when a certain thing is accessed. In constructing this graph, we integrate the spatial and temporal information to capture periodic patterns between locations and timestamps for improved performance. Then, we perform random walks with restart on both graphs to obtain pairwise relevance \( R_m \) and \( R_u \) respectively, and sum them up to get overall pair correlation \( R = (R_m + R_u)/2 \), which indicates thing-thing correlations. More details can be found in Part I.

Similar to modeling social networks, things correlation matrix is decomposed to thing-based latent features vectors \( v_j \in \mathbb{R}^{1 \times n} \) and factor-based latent feature vectors \( v'_j \in \mathbb{R}^{1 \times n} \) on analyzing the things correlation graph \( G_t \). The conditional probability of \( t_{jj'} \) relies on things latent factors and can be denoted as:

\[
t_{jj'} \sim Pr(t_{jj'}|v_j^T v'_j; \sigma_t)
\]

where \( v_j \sim \mathcal{N}(0, \sigma^2_v) \), \( v'_j \sim \mathcal{N}(0, \sigma^2_v) \) \( (8.6) \)

8.3.3 Encoding User-Things Interactions

User-Things interactions \( y_{ij} \) are embodied by the usage frequency of thing \( i \) by user \( j \) in a certain timespan. We can map the usage frequency to interval \([0, 1]\) by using function \( f(x) = \frac{x - y_{min}}{y_{max} - y_{min}} \) without loss of generality, where \( y_{max} \) and \( y_{min} \) are the maximum and minimum usage values respectively. The dyadic relationship between a user and a thing does not only depend on their latent factor \( U^T V \), whose vulnerability is that it makes use of past interactions and can not handle brand new things well, i.e., cold start problem. To tackle this issue, we use the explicit features directly by profiling users observable features \( x_i \in \mathbb{R}^c \) (i.e., age, gender, location etc.) and things observable features \( z_j \in \mathbb{R}^d \) (i.e., textual description of things functionalities,
things contextual information etc) [135]. \(c\) and \(d\) are the dimensionality of users observable features and things observable features respectively. The dyadic relationship (thing usage value) depends on not only the inner product of latent factors of users and things, but also their observable features. Things usage value \(y_{ij}\) can be defined as the following conditional probability:

\[
y_{ij} \sim Pr(y_{ij} | u_i, v_j, x_i, z_j, \sigma_y^2)
\] (8.7)

We adopt the bilinear product to specify the similarity between user observable features and thing observable features [26]. The pairwise similarity between \(x_i\) and \(z_j\) can be denoted as:

\[
r_{ij} = w^T (x_i \otimes z_j)
\] (8.8)

where \(w\) is a column vector of entries \(\{w_{mn}\}\), and \(x_i \otimes z_j\) denotes the Kronecker product of \(x_i\) and \(z_j\). So Equation 8.8 can be rewritten as:

\[
r_{ij} = x_i^T W z_j
\] (8.9)

where matrix \(W \in \mathbb{R}^{m \times n}\) is a weight coefficients capturing pairwise associations between user \(i\)’s explicit feature vector and thing \(j\)’s explicit feature vector. The thing usage value depends on both the inner product of user and thing latent factors and the bilinear product of user observable features and thing observable features. Equation 8.7 can be reformulated as:

\[
y_{ij} \sim Pr(y_{ij} | u_i^T v_j + r_{ij}, \sigma_y^2)
\] (8.10)

where \(w \sim \mathcal{N}(0, \sigma_w)\).
Chapter 8. Exploring Things Recommendation in Internet of Things

8.3.4 Model Learning

Given a training dataset for $\mathcal{O} = \{ \mathcal{O}_y, \mathcal{O}_s, \mathcal{O}_t \}$, the joint posterior probability of model parameters $\Sigma = \{ w, \sigma_w, \sigma_u, \sigma_u', \sigma_v, \sigma_v' \}$ can be obtained through Bayes’ theorem:

$$Pr(\Sigma | \mathcal{O}) \propto Pr(\mathcal{O} | \Sigma) Pr(\Sigma)$$  \hspace{1cm} (8.11)

Maximizing Equation 8.11 can be converted to minimizing the negative logarithm of $Pr(\mathcal{O} | \Sigma) Pr(\Sigma)$ via:

$$\min_{\Sigma} L(\Sigma) = \min_{\Sigma} \lambda_y \sum_{y \in \mathcal{O}_y} \ell(y_{ij}, u_i^T v_j + r_{ij}) + \lambda_s \sum_{u' \in \mathcal{O}_s} \ell(s_{i,i'}, u_i^T u'_i) + \lambda_t \sum_{j,j' \in \mathcal{O}_t} \ell(t_{jj'}, v_j^T v_{j'})$$ \hspace{1cm} (8.12)

$$+ \lambda_w \|w\|^2 + \lambda_u \|u\|^2 + \lambda_u' \|u'\|^2 + \lambda_v \|v\|^2 + \lambda_v' \|v'\|^2$$

where $\ell \cdot$ is a loss function (we adopt the most widely used $\ell_2$ loss), and $\lambda = \{ \lambda_y, \lambda_s, \lambda_t, \lambda_w, \lambda_u, \lambda_u', \lambda_v, \lambda_v' \}$ are trade-off parameters.

A gradient descent process can be implemented to solve the parameters. Given a training dataset $\{ y_{ij} \}$, the objective function in Equation 8.12 can be found by performing gradient descent in $u_i, v_j$ and $w_{mn}$.

$$u_i \rightarrow u_i - \delta \times \left( \frac{\partial \ell}{\partial u_i} (y_{ij}, u_i^T v_j + r_{ij})v_j + u_i^T u_i \right)$$

$$v_j \rightarrow v_j - \delta \times \left( \frac{\partial \ell}{\partial v_j} (y_{ij}, u_i^T v_j + r_{ij})u_i + v_j^T v_j \right)$$

$$w \rightarrow w - \delta \times \left( \frac{\partial \ell}{\partial w} (y_{ij}, u_i^T v_j + r_{ij})u_i v_j + w^T w \right)$$ \hspace{1cm} (8.13)

$$u'_{i'} \rightarrow u'_{i'} - \delta \times \left( \frac{\partial \ell}{\partial u'_{i'}} (s_{i,i'}, u_i^T u'_{i'})u_i + u'^T u'_{i'} \right)$$

$$v'_{j'} \rightarrow v'_{j'} - \delta \times \left( \frac{\partial \ell}{\partial v'_{j'}} (t_{jj'}, v_j^T v'_{j'})v_j + v'^T v'_{j'} \right)$$

where $\delta$ is the learning rate. After obtaining the optimal parameters $\Sigma^*$, we can use them to predict the given testing data $\{ \hat{i}, \hat{j}, y_{i\hat{j}} \}$.
where $c$ and $d$ are the dimensionality of users’ explicit features $x$ and things explicit features $z$.

### 8.4 Experiments

In this section, we report our experimental results. We conducted two experiments to evaluate the performance of our approach. The first experiment is to compare our approach with other state-of-the-art methods. The second experiment is to demonstrate the impact of social networks and things correlations respectively.

#### 8.4.1 Dataset Description

Since Internet of Things (IoT) is still relatively new, it is hard to find large-scale data for our experiments. We built up a comprehensive platform named T-Mine to deploy Internet of Things applications and built a ground truth dataset to validate our approach (details can be found in Part I).

A thing usage event is generated when a person interacts with a particular thing. Let $O = \{o_1, ..., o_n\}$, $U = \{u_1, ..., u_m\}$, $Ts = \{ts_1, ..., ts_p\}$ and $Loc = \{loc_1, ..., loc_q\}$ represent the set of things, users, timestamps and locations, respectively. A usage event of a thing $o_i$, denoted by $h \in H = \{h_1, ..., h_i\} = \{< o, u, ts, loc > | o \in O \land u \in U \land ts \in Ts \land loc \in Loc\}$, indicates that user $u$ used a particular thing $o$ located in a specific location $loc$ at a particular time $ts$. We can calculate the usage frequency for every objects from things usage events.

Table 8.1 shows the statistics of six categorical things of the dataset. Scale refers to the usage frequency between the most frequent used things and the least frequent.
used things in each category. For example, in *Cooking*, one of the least frequent used thing is the blender (48), while one of the most frequent used thing is the fridge (2336, the door open/close times of the fridge). All scale values are processed to make them in [0,1] as discussed in Section 8.3.3.

<table>
<thead>
<tr>
<th>No.</th>
<th>Category</th>
<th># Things</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Entertainment</td>
<td>28</td>
<td>47 - 1028</td>
</tr>
<tr>
<td>2</td>
<td>Office</td>
<td>20</td>
<td>511 - 1790</td>
</tr>
<tr>
<td>3</td>
<td>Cooking</td>
<td>25</td>
<td>48 - 2336</td>
</tr>
<tr>
<td>4</td>
<td>Transportation</td>
<td>11</td>
<td>31 - 870</td>
</tr>
<tr>
<td>5</td>
<td>Medicine/Medical</td>
<td>10</td>
<td>22 - 89</td>
</tr>
<tr>
<td>6</td>
<td>House Appliances</td>
<td>33</td>
<td>37 - 2680</td>
</tr>
</tbody>
</table>

8.4.2 Performance Metrics

We adopt Mean Absolute Error (MAE) to measure the accuracy of our approach. MAE calculates the average of absolute difference between predicted usage values and actual values as the following:

$$ MAE = \frac{\sum_{ij} |y_{ij} - \hat{y}_{ij}|}{n} $$  \hspace{1cm} (8.15)

where $y_{ij}$ is the actual thing’s usage values between user $i$ and thing $j$, $\hat{y}_{ij}$ is the predicted thing’s usage value, and $n$ is the number of the predicted thing’s usage values. The lower the MAE, the better the performance. It should be noted that, since our data set is not large, it is hard for us to do the experiments on each category. In our experiments, we calculated MAE based on the overall MAE on prediction things usage value across all categories.
8.4.3 Performance Comparison

In this section, we compare the prediction accuracy of our proposed approach based on fusing social networks and things correlations (FST) with some state-of-the-art approaches based on probabilistic factor analysis: Probabilistic Matrix Factorization (PMF) [104], SoRec [81] and SVD++ [64].

- Probabilistic Matrix Factorization (PMF) is briefly defined as:

  \[ y_{ij} \sim Pr(u_i^T v_j, \sigma_y) \]  
  (8.16)

  where \( u_i \) and \( v_j \) are the low dimensional factors learned from user-item interactions.

- SoRec integrates users’ social network structure and the user-item interaction matrix. The integration is based on the probabilistic factor analysis through the shared user latent feature space \( u_i \), by learning the low-rank user latent feature space \( u_i \) and \( u_i' \) on social network, and the item latent feature space \( v_j \) on user-item interaction matrix. It can be defined as:

  \[ s_{ii'} \sim p(s_{ii'} | u_i^T u_i', \sigma_s) \]

  \[ y_{ij} \sim p(y_{ij} | u_i^T v_j, \sigma_y) \]

  (8.17)

- SVD++ combines the neighborhood models and latent factor models together.

  \[ y_{ij} \sim p(y_{ij} | \hat{u}_i^T v_j, \sigma_y) \]

  (8.18)

  where \( \hat{u}_i = \sum_{i' \in N_i} \omega_{ii'} u_{i'} \), and \( N_i \) refers to neighbor of user \( i \) and \( \omega_{ii'} \) is the similarity between user \( i \) and user \( i' \).
Table 8.2: Comparison with PMF, SVD++, SoRec on all categories

<table>
<thead>
<tr>
<th>Training Data</th>
<th>10%</th>
<th>20%</th>
<th>50%</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Factors</td>
<td>5</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>PMF</td>
<td>0.8635</td>
<td>0.8544</td>
<td>0.8245</td>
</tr>
<tr>
<td>SVD++</td>
<td>0.8425</td>
<td>0.8311</td>
<td>0.8004</td>
</tr>
<tr>
<td>SoRec</td>
<td>0.8103</td>
<td>0.7978</td>
<td>0.7872</td>
</tr>
<tr>
<td>FST</td>
<td>0.7903</td>
<td>0.7746</td>
<td>0.7712</td>
</tr>
</tbody>
</table>

This experiment evaluated our approach, in particular its capability in handling the cold start problem, which refers to providing accurate prediction when some users only use few things or even have no usage historical records at all. In order to verify the capability of our approach on predicting usage value of things that have not been used, we randomly selected and marked off \( p\% \) of data (\( p = 10, 20 \) and \( 50\) ) from our dataset as training data and different number of latent factors (5 and 10) to test all the methods. The experimental results are shown in Table 8.2.

From the table, it is clear that our approach outperforms other methods on different training ratios and different number of factors, especially when the training data is small, i.e., when the training ratio is 10%. The reason lies in PMF is a pure probabilistic factor model. Relying heavily on user-thing usage matrix, it can not deal with the circumstance where little interactions information is available. SoRec works better than PMF and SVD++ because of its aggregation of user-user internal information (social links). Our approach not only incorporates users and things internal information, but also defines the explicit features (i.e., content) for users (e.g., users profile) and things (e.g., description of things functionalities), which makes our approach performing better when there is a cold start problem. The experimental result further demonstrates the effectiveness on improving the recommendation accuracy by incorporating things correlations.
8.4.4 Impact of Things Correlations and Social Influence

This experiment evaluated the impact of users’ relations and things correlations to our model. To do so, we excluded the things correlations information for our model, namely FST/T, and the social relations from the model (FST/S) respectively. FST/T is defined as:

\[
y_{ij} \sim Pr(y_{ij}|u_i^T v_j + r_{ij}, \sigma_y)
\]
\[
s_{ii'} \sim Pr(s_{ii'}|u_i^T u_{i'}^T)
\]

And FST/S is defined as:

\[
y_{ij} \sim Pr(y_{ij}|u_i^T v_j + r_{ij}, \sigma_y)
\]
\[
t_{jj'} \sim Pr(t_{jj'}|v_j^T v_{j'}^T)
\]

FST/T is similar to SoRec in [81], but it is imported with the explicit features of users and things. Besides, we excluded the social influence factors, namely FST/S, which has the same structure as FST/T since our model FST holds a symmetrical structure. We implemented FST/T, FST/S and FST, and conducted experiments to study their performance. We varied the training ratio as 10%, 20% and 50%, and the number of factors are 5 and 10 respectively. Figure 8.3 shows the experimental results.

The results show that the performance of FST/T drops down when we take out the things correlations from FST. This proves our intuition that things’ correlations do affect users decision making process and behavior pattern when they use services offered by different things. Another finding is that FST/S consistently performs better than FST/T. In other words, things correlations has more influence than users’ social relations. Comparing with traditional Internet resource like images or documents, physical things have bigger impact on users’ behavior because of their closeness with people and its non-duplicability. This is indeed a unique feature of Internet of Things.
8.5 Related Work

In this section, we provide an overview about collaborative filtering approaches, which are closely related to our model.

Collaborative techniques exploit historical interactions of user behavior for future prediction based on either neighborhood based [106, 29, 74] or latent factor based methods [63, 52, 104]. The idea behinds neighborhood-based methods is that interactions between users and items can be inferred from the observation of users or items neighborhood. It predicts very well in learning the locality of dependencies, but does not explain the global relationships in the user-item interactions. The idea behind la-
tent factor models is that preferences of a user are determined by a small number of latent factors, and it can learn the informative latent feature space but fails in capturing the local dependency of user or item neighborhood. Multiple unified models are proposed to combine neighborhood-based methods and latent factor based methods together. For example, [64] propose an approach combining neighborhood-based CF and latent factor model together, and performance is significantly improved. [103] proposes a probabilistic matrix factorization framework, which scales linearly with the number of observations and performs well on the large and sparse dataset. However, all of them can not handle the cold-start problem very well. [1] addresses the cold-start problem by integrating explicit features of users and items into latent factors learning process. They develop a regression-based latent factor model for rating prediction, which uses features for factor estimation. In their methods, the user and item latent factors are estimated through independent regression on user and item features, and the recommendation is calculated from a multiplicative function on both user and item factors.

However, these research efforts assume users are independent or unrelated to each other. This assumption does not work well in context of social networks where users social interactions have big impact on recommending process. Most recent research work focus on exploiting the information of users’ connections for recommendation. Many approaches have been developed to integrate users’ social information in recommendation. [55] proposes a social recommendation framework based on probabilistic matrix factorization via employing user social networks and user-item ratings. [134] designs a joint friendship and interest propagation model, where the user-item interest network and the user-user friendship network (side information on users) are jointly modeled through latent user and item factors. [159] proposes a kernel-based probabilistic matrix factorization, which incorporating external information into the matrix factorization process via assuming latent factors are in Gaussian distribution. [44] de-
Chapter 8. Exploring Things Recommendation in Internet of Things

develops a unified model for CF based on graph regularized weighted nonnegative matrix factorization. They adopt user demographic and item genre information to construct neighborhood graphs and incorporated user and item graphs in weighted nonnegative matrix factorization. [81] proposed a matrix factorization method to exploit the social network information. Compared with these research efforts, we construct the things correlation graph indicating global connections and similarities between things, and integrate the graph with users’ social relation graph to learning the latent factors simultaneously. The technical details are given in next sections.

8.6 Conclusion

Recommending the right thing to use at the specific temporal and spatial location is a fundamental concern in the emergent Internet of Things research. The work presented in this chapter is a continuing exploration of our previous work in [137]. In particular, we have shown that both users’ relations and things correlations have synergy effect on things recommendation process. We propose a probabilistic matrix factorization based model integrating users’ social relations (i.e., friendship) and things correlations together, where simultaneous latent factors are learned from unrevealing three dyadic relationships, including user-user, user-thing and thing-thing relationships. We also integrate the content and other additional information of users and things to cope with the cold-start problem. The experiments demonstrate the feasibility and effectiveness of our model.

In this work, we do not consider the dynamic features of things. In real situation, physical things are more dynamic compared with traditional web resources. Some of those dynamic features are availability, statefulness and its changing attributes (i.e., geographical information, status). In the future work, we plan to improve our model that can adaptively propagate up-to-date information from things correlations network and
make more accurate recommendations. The things states can be dynamically mon-
itored and updated in the recommending process. This work can benefit from our
previous things availability tracking approach proposed in Chapter 7 of Part II.
Chapter 9

Unified Collaborative Service Recommendation Combined with Content-based Features

As shown in Chapter 3 of Part I, content-based feature is a crucial resource for various things mining tasks, e.g., things annotation. Since things are exposed as services and resources via the Web service interfaces (i.e., RESTful web services), the content features of things can be extracted based on generic natural language processing methods (i.e., tf/idf).

In this chapter, we further propose a novel approach for things recommendation by unifying collaborative filtering recommendation (i.e., people-things historical interactions) and content-based features in a normative manner. Our approach exploits the advantages of both techniques that enables more accurate recommendations with a rich variety. More specifically, our approach is built on a three-way aspect model [96] that directly represents unobservable user preferences as a set of latent variables, which can be statistically estimated using algorithms such as expectation maximization (EM) [124]. In a nutshell, the main contributions of our work are as the following:

- We analyze tasks on service recommendation and identify three main require-
ments that are important for designing effective things recommender systems.

- We propose a novel hybrid approach that combines collaborative filtering and semantic content-based methods for service recommendation. Our approach exploits a *three-way aspect* model that simultaneously considers the similarities of users and semantic content of services. User preferences are represented using a set of latent variables that can be statistically estimated. We further develop two strategies to specifically deal with the overfitting problem caused by data sparsity.

- We conduct extensive experiments using real-world services to verify the proposed approach. A large scale dataset [156] consisting 5,825 services are carefully examined and 3,693 live services are selected and used in the experiments. The experimental results show that our approach achieves better recommendation performance than the conventional collaborative filtering and content-based methods applied separately.

### 9.1 Motivations and Challenges

With the increasing adoption and presence of Internet of Things, more things are poured into Internet and provide various services. This calls for novel approaches for efficient and effective service recommendation, which is a critical issue in many further practical applications in Internet of Things, such as things discovery, search and composition [156, 142, 53, 3].

Service recommendation is the process of automatically identifying the usefulness of services and proactively recommending services to end users. We can also view service recommendation as the process of service selection augmented with end user behavior analysis to achieve relevant and accurate service suggestions. Over
Chapter 9. Unified Collaborative Service Recommendation Combined with Content-based Features

the last few years, many service discovery approaches have been proposed [16, 154, 31] and several services search engines have emerged such as WebServiceList\(^1\), XMethods\(^2\), and ProgrammableWeb\(^3\). These search engines largely exploit keyword-based search techniques and are insufficient to fully describe the functionalities of services. Furthermore, accommodation for non-functional characteristics such as quality of service (QoS) of services during the service selection and recommendation are very limited [20]. In a recent work by Zheng et al. [156, 154], a service search engine is designed and developed that ranks services not only by functional similarities to a user’s query, but also by non-functional QoS characteristics of services.

The main goal of our work is to advance the current state-of-the-art on service recommendation. More specifically, our work is inspired by the following observations and further developed for recommending services offered by things in Internet of Things situation. To find desirable services by using service search engines, a user normally has to supply the queries and often at a loss as to what queries are appropriate (e.g., which keywords should be used, what values should be set for a QoS attribute). Another problem is that services that do not satisfy the user’s search query are completely excluded from the recommendation list. It is therefore desirable that a recommendation system selects probably-preferred services by estimating user preferences without requiring users to explicitly specify those preferences.

In the last few years, two main service recommendation techniques have been proposed: collaborative filtering and content-based recommendation. Collaborative filtering [22, 51, 156, 63] is a technique that has been widely used for recommending items (services offered by things in our case) to a given user by considering other similar users’ ratings on the items. For instance, suppose that a user likes services \(s_a\) and \(s_b\).

---
\(^1\)http://www.webservicelist.com
\(^2\)http://www.xmethods.net
\(^3\)http://www.programmableweb.com
If there are many other users who like $s_a$ and $s_b$ also like service $s_c$, then service $s_c$ should probably be recommended to that user. Although this technique is effective, one big problem is that services without a considerable set of user interactions (e.g., newly deployed services) cannot be recommended, which is also known as the cold start problem. On the other hand, content-based methods [33, 15, 76] recommend services based on the similarity of user preferences and the descriptive information of services (e.g., functionalities). Newly-deployed services can be recommended by this technique. Unfortunately, associating user preferences with service content is not a trivial task and very few solutions have been proposed. In current service search engines, queries that represent user preferences are typically prepared by users.

9.2 Requirements in Service Recommendation

Accuracy is an important metric when assessing a recommender system. However, there is an increasing awareness that good accuracy alone does not necessarily give users an effective and satisfying experience [50]. In this chapter, we identify three major requirements in order to conduct an effective service recommendation task:

- **High recommendation accuracy.** A good recommender system should recommend more relevant services and fewer irrelevant ones, particularly in the situations where required information are not available (e.g., missing QoS of some services) [156].

- **Recommendation serendipity.** Recommending services that are already known to a user can be found unsatisfactory or meaningless. If the recommended services are not familiar to the user, the chances of finding new services that match the user’s requirements would increase [50].
Chapter 9. Unified Collaborative Service Recommendation Combined with Content-based Features

- **Recommended newly deployed services.** Overcoming the cold-start problem not only enables users to find newly-deployed services, but also enhances the recommendation serendipity.

### 9.2.1 Problem Description

Let \( \mathcal{U} = \{u|1, \ldots, N_u\} \) and \( \mathcal{S} = \{s|1, \ldots, N_s\} \) be the set of users and services in a recommender system respectively. Here \( N_u \) and \( N_s \) are the number of the users and the services. The relationship between service users and services can be denoted by a user-item matrix \( \mathcal{R} \):

\[
\mathcal{R} = \{r_{u,s}|1 \leq u \leq N_u, 1 \leq s \leq N_s\}
\]

An entry in \( \mathcal{R} \), denoted by \( r_{u,s} \), represents a vector of QoS values (e.g., response time) of service \( s \) that is observed by the service user \( u \). When user \( u \) has not invoked a service \( s \), \( r_{u,s} = \emptyset \). It should be noted that most entries in \( \mathcal{R} \) are empty in real-world applications. The reason is that the number of services invoked by each user is usually very small. Collaborative filtering based approaches use \( \mathcal{R} \) for service recommendation.

In content-based recommendation, the content of each service is represented as a vector of several features extracted from e.g., the WSDL file and the short service description. Let \( \mathcal{N}_f \) be the number of features and \( c_{s,t} \) be the value of the \( t^{th} \) feature of service \( s \). By collecting all the feature vectors, we can have the content matrix \( \mathcal{C} \) as:

\[
\mathcal{C} = \{c_{s,t}|1 \leq s \leq N_s, 1 \leq t \leq \mathcal{N}_f\}
\]

Given a target user \( u \), content-based approaches use \( \mathcal{C} \) for service recommendation. Unlike collaborative-based approaches, they do not consider the results from other users. We will discuss further the two different kinds of approaches in the following sections.
9.3 The Unified Service Recommendation Model

To meet the three requirements described in Section 9.2, we propose a unified approach that combines collaborative filtering technique and content-based approach. To achieve this, it is necessary to reflect both rating and content data in modeling of user preferences. Unfortunately, user preferences are only indirectly represented and observable data such as ratings or content (e.g., semantic descriptions) do not completely reflect the preferences.

To solve the problem, we propose to use a probabilistic model that associates ratings and content data with newly-introduced variables that represent user preferences. In particular, our work is inspired by a three-way aspect model presented in [96]. This model has a set of latent variables that directly describe substantial preferences, which cannot be observed directly. The preferences are statistically estimated using expectation maximization (EM) [28] that thereafter contribute to better recommendation. In the rest of this section, we will describe how to adapt this model for service recommendation.

9.3.1 Model Description

The graphical representation of the three-way aspect model for service recommendation can be found in Figure 9.1. The model includes four components: a user set \( U = \{u_1, u_2, \ldots, u_{N_u}\} \), a service set \( S = \{s_1, s_2, \ldots, s_{N_s}\} \), semantic content of services \( C = \{c_1, c_2, \ldots, c_{N_c}\} \) where \( c_i \) is a semantic description of service, and a set of latent variables \( Z = \{z_1, z_2, \ldots, z_{N_z}\} \) that governs the recommendation process, e.g., users’ latent preferences.

The model captures a three-way co-occurrence data among users, services, as well as the content of services in the form of semantic descriptions. An observation is
Chapter 9. Unified Collaborative Service Recommendation Combined with Content-based Features 169

Figure 9.1: Graphical representation of our probabilistic model

typically a triple \((u, s, c)\) that corresponds to an event where a user \(u\) accesses a service \(s\) that contains a semantic service description \(c\). In the three-way aspect model, observation data is associated with one of the latent variables \((z_i \in Z)\). The latent variables represent user latent preferences to services, e.g., preferences on functionalities or non-functionalities (QoS values). It is assumed that users, services, and semantic descriptions are independent in the model. It is also worth noting that the aspect model allows multiple semantic descriptions per user, unlike most clustering methods that assign each user with a single class.

In the context of service recommendation, an event of a user \(u \in U\) accessing a service \(s \in S\) containing semantic description \(c \in C\), is considered to be associated with one of the latent variables \(z \in Z\). Conceptually, users choose (latent) topics \(z\),
which in turn “generates” both services and their content description. Therefore, a latent variable in this new model is not only associated with a distribution of services but also a distribution of semantic service content. The joint probability distribution \( P_r(u, s, c, z) \) over user set \( U \), latent topic variables \( Z \), service set \( S \) and service content \( C \) is given by

\[
P_r(u, s, c, z) = P_r(u)P_r(z|u)P_r(s, c|z)
\] (9.3)

Since we consider the distribution of \( s \) and \( c \) are independent in our model, we can have \( P_r(s, c|z) = P_r(s|z)P_r(c|z) \). The above equation can be rewritten as:

\[
P_r(u, s, c, z) = P_r(u)P_r(z|u)P_r(s|z)P_r(c|z)
\] (9.4)

An equivalent specification of the joint probability distribution that treats users and items symmetrically is:

\[
P_r(u, s, c, z) = P_r(z)P_r(u|z)P_r(s|z)P_r(c|z)
\] (9.5)

Marginalizing out \( z \), we obtain the joint probability distribution \( P_r(u, s, c) \) over \( U \), \( S \), and \( C \) as the following:

\[
P_r(u, s, c) = \sum_z P_r(z)P_r(u|z)P_r(s|z)P_r(c|z)
\] (9.6)

This model has a set of parameters \( P_r(z), P_r(u|z), P_r(s|z) \) and \( P_r(c|z) \), which for simplicity is represented as \( \theta \). The model parameters are learned by mining the user-service history data \( H = \{ <u, s, c> \} \). One way to learn \( \theta \) is to maximize the log-likelihood of history data which is:

\[
\mathcal{L}(\theta) = \sum_{<u,s,c> \in H} n(u, s, c) \log(P_r(u, s, c|\theta))
\] (9.7)
In our work, we adopt the EM algorithm [28] to find a local maximum of the log-likelihood of the training data, the detailed model learning process will be presented in Section 9.3.2.

After the model is learned, the inference of services can be ranked for a given user according to $Pr(s|u) \propto \sum_c Pr(u, s, c)$, i.e., according to how likely it is that the user will invoke the corresponding service. Services with high $Pr(s|u)$ that the user has not yet invoked are good candidates for recommendation, which addresses the requirement of recommendation serendipity raised in Section 9.2. Since the model considers the content of services, the cold-start problem can also be solved (i.e., newly-deployed services can be recommended).

### 9.3.2 Model Learning

Let $n(u, s, c) = r(u, s) \times n(s, c)$, where $n(u, s, c)$ indicates how much a user $u$ prefers the semantic descriptor $c$ in service $s$, $r(u, s)$ is the rating score of user $u$ for service $s$, and $n(s, c)$ is the number of times semantic descriptor $c$ occurs in service $s$. Given training data of this form, the log likelihood $\mathcal{L}$ of the data is:

$$\mathcal{L}(\theta) = \log \prod_{<u,s,c> \in \mathcal{H}} Pr(u, s, c|\theta) \quad (9.8)$$

which can be rewritten as:

$$\mathcal{L}(\theta) = \sum_{<u,s,c> \in \mathcal{H}} n(u, s, c) \log(Pr(u, s, c)|\theta) \quad (9.9)$$

One way to learn $\theta$ is to maximize the log-likelihood of the history data. However, directly maximizing $\mathcal{L}(\theta)$ is hard. The EM algorithm applies an iterative method to improve model parameters. Equation 9.9 can be derived as:
Chapter 9. Unified Collaborative Service Recommendation Combined with Content-based Features

\[ \mathcal{L}(\theta) = \sum_{(u,s,c) \in \mathcal{H}} n(u, s, c) \log(Pr(u, s, c|\theta)) \]

\[ = \sum_{(u,s,c) \in \mathcal{H}} \log Pr(u, s, c|\theta) \]

\[ = \sum_{(u,s,c) \in \mathcal{H}} \log \left( \sum_{z} Pr(z|u, s, c, \theta^{(t)}) \frac{Pr(u, s, c, z|\theta)}{Pr(z|u, s, c, \theta^{(t)})} \right) \]

\[ \geq \sum_{(u,s,c) \in \mathcal{H}} \sum_{z} Pr(z|u, s, c, \theta^{(t)}) \log \left( \sum_{z} Pr(z|u, s, c, \theta^{(t)}) \right) \]

\[ \frac{Pr(u, s, c, z|\theta)}{Pr(z|u, s, c, \theta^{(t)})} \triangleq Q(\theta|\theta^{(t)}) \]

Therefore, instead of maximizing \( \mathcal{L}(\theta) \) directly, the EM algorithm tries to find the model parameters \( \theta^{(t+1)} \) to maximize \( Q(\theta|\theta^{(t)}) \). So:

\[ \theta^{t+1} = \arg \max \{ Q(\theta|\theta^{(t)}) \} \]

\[ = \arg \max_{\theta} \left\{ \sum_{(u,s,c) \in \mathcal{H}} \sum_{z} Pr(z|u, s, c, \theta^{(t)}) \log Pr(u, s, c, z|\theta) \right\} \]

\[ = \arg \max_{\theta} \left\{ \sum_{(u,s,c) \in \mathcal{H}} \mathbb{E}_{z|u,s,c,\theta^{(t)}} \{ \log Pr(u, s, c, z|\theta) \} \right\} \]

\[ (9.11) \]

At this point, we can use the EM algorithm to solve Equation 9.11 with training dataset. In particular, the E step and M step are iterated alternately until the log-likelihood \( \mathcal{L} \) converges to a local maximum. It should be noted that both content and collaboration data can influence recommendations. The relative weight of each type of data depends on the nature of the given data for training.
The E-step is used to obtain the posterior probabilities in Equation 9.10 by calculating \( Pr(z|u, s, w, \theta^t) \), where the model parameters \( \theta^t \) are known in this step:

\[
Pr(z|u, s, c, \theta^t) = \frac{Pr(z)Pr(u|z)Pr(s|z)Pr(c|z)}{\sum_z Pr(z)Pr(u|z)Pr(s|z)Pr(c|z)} \tag{9.12}
\]

In the M-step, we need to find new model parameters to maximize the expected log-likelihood found in the E-step, since

\[
\mathcal{L}(\theta) = \log Pr(u, s, c, z|\theta)
= \log Pr(u|z) + \log Pr(s|z) + \log Pr(c|z) + \log Pr(z) \tag{9.13}
\]

So, the maximization to the model parameters \( \theta^{t+1} = \{Pr(u|z), Pr(s|z), Pr(c|z), Pr(z)\} \) can be obtained by maximizing the expectation with respect to \( \theta \):

\[
Pr(u|z) \propto \sum_{s,c} n(u, s, c) Pr(z|u, s, c)
Pr(s|z) \propto \sum_{u,c} n(u, s, c) Pr(z|u, s, c)
Pr(w|z) \propto \sum_{u,s} n(u, s, c) Pr(z|u, s, c)
Pr(z|z) \propto \sum_{u,s,c} n(u, s, c) Pr(z|u, s, c) \tag{9.14}
\]

After the model is learned, the inference of Web services can be ranked for a given user according to \( Pr(s|u) \propto \sum_c Pr(u, s, c) \), i.e., according to how likely it is that the user will invoke the corresponding Web service. Web services with high \( Pr(s|u) \) that the user has not yet invoked are good candidates for recommendation. This addresses the requirement of recommendation serendipity discussed in Section 9.2. In addition, since the model considers the content of Web services, the cold-start problem can also be solved (i.e., newly-deployed Web services can be recommended).
It is noted that the likelihood needs to iterate over all possible data connections of each data sample, which might be computationally expensive. For example, in the E-step of the algorithm, we need to calculate the expectation posterior distribution of $Pr(z|u, s, c)$, given the current estimated parameters $Pr(u|z)$, $Pr(s|z)$ and $Pr(c|z)$. This computation is heavy due to standard least-squares computation, which is estimation of the regression coefficients of the factors on the variables assuming that the current estimated $Pr(u|z)$, $Pr(s|z)$ and $Pr(c|z)$ are found from the M-step. The time complexity of implementing the EM algorithm is $O(N \cdot |c| \cdot \mathcal{N}_z)$, where $\mathcal{N}$ is the size of the training dataset, $|c|$ is the length of descriptive vectors of services and $\mathcal{N}_z$ is the number of latent variables. Another concern is that although the EM algorithm is guaranteed to be stable and to converge to a local maximum value of the estimated likelihood which depends on the initial data, there is no guarantee that this value is globally maximum.

In reality, the user-service interaction matrix could be very sparse, in other words, the users only invoke a few part of services. However, when data is extremely sparse, which is typical in many real-world applications, the EM algorithm will suffer from overfitting, i.e., poor generalization. We will discuss two strategies in the next section that can effectively increase the data density, which in turn improves the learning performance of the EM algorithm.

### 9.4 Dealing with Data Sparsity

In this section, we describe two strategies in overcoming the overfitting problem caused by sparse data. The first strategy is to preprocess data matrix using a data smoothing technique. The second strategy is to modify the three-way aspect model by eliminating services from direct participation in the model.
9.4.1 Data Smoothing

The idea of the data smoothing strategy to addressing the overfitting problem with sparse data is to smooth the data matrix based on the similarities between services. The intuition behind is as follows. Consider a user $u$ who has invoked service $s_i$. Assume that another service $s_j$ has not been invoked by $u$, but $s_i$ and $s_j$ are very similar in content (e.g., both services share many service attributes). Informally, if the content similarity function (see Equation 5), $q(s_i, s_j)$, yields 0.85, we could believe that there is a 85% chance that user $u$ has actually invoked service $s_j$, even though the recommender system does not know it.

Based on this reasoning, we propose to preprocess the original user-item matrix $\mathcal{R}$ by filling in some of the empty entries with the average similarities above a certain threshold between a service and all other services invoked by user $u$. Clearly, when the threshold is bigger, the data matrix will become sparser (i.e., less empty entries will be replaced). On the other hand, if the threshold is smaller, the data matrix will become less sparse (i.e., more empty entries will be replaced). By setting appropriate threshold, the density of the data matrix (i.e., the fraction of non-zero entries) can be effectively increased, as a result of the data smoothing.

9.4.2 Implicit User-Descriptor Aspect Model

A service contains multiple semantic service descriptors and a service descriptor is contained in many services. Another method to overcome the overfitting problem due to sparsity is to propose a model where co-occurrence data points represent events that correspond to users looking for service descriptors in a service, i.e., $(u, c)$. This is different from the model in Section 9.3 where the event corresponds to a user accessing a service with service descriptors, $(u, s)$. 
This modified aspect model produces estimates of conditional probabilities $Pr(u|z)$ and $Pr(c|z)$, as well as the latent variable priors $Pr(z)$. $Pr(u|c)$ can be calculated using:

$$Pr(u, c) = \sum_z Pr(z)Pr(u|z)Pr(c|z)$$

(9.15)

However, the task of a recommender system is to recommend services that have the highest estimating probabilities $Pr(s|u)$ for a given user $u$. We can solve this problem by treating a service as a bag of service descriptors: the probability of a service is the product of the probabilities of the semantic service descriptors it contains adjusted for different service description lengths with geometric mean:

$$Pr(s, u) \propto \left( \prod_i Pr(c_i, u) \right)^{1/|c|}$$

(9.16)

where $c_i$ are semantic service descriptors in a service $s$ and $|c|$ is the length of the service description of $s$. Conditional probabilities $Pr(c_i, u)$ follow directly from the model:

$$Pr(c_i, u) = \frac{Pr(u, c_i)}{\sum_c Pr(u, c)}$$

(9.17)

### 9.5 Performance Evaluation

This section focuses on reporting the performance study of our proposed hybrid approach for service recommendation, including three experiments: i) comparing our hybrid approach with the conventional methods including collaborative filtering and content-based recommendation, ii) studying the sensitivity of the hybrid approach under different markoff ratios and latent variables, and iii) investigating the scalability of our approach for large-scale, distributed datasets.
Table 9.1: Dataset Statistics

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of Users</th>
<th>Number of Web Services</th>
<th>User-Service (Response Time) Matrix Density</th>
<th>User-Service (Throughput) Matrix Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Dataset</td>
<td>339</td>
<td>5825</td>
<td>$5.11 \times 10^{-2}$</td>
<td>$7.26 \times 10^{-2}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of Users</th>
<th>Number of Web Services</th>
<th>User-Service (Rating) Matrix Density</th>
<th>Average Number of Semantic Descriptors for Each Service</th>
<th>Average QoS Ratings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processed Dataset</td>
<td>572</td>
<td>3693</td>
<td>$7.67 \times 10^{-2}$</td>
<td>12.79</td>
<td>3.84</td>
</tr>
</tbody>
</table>

9.5.1 Dataset Setup

To perform reliable experiments, it is ideal to use large-scale real world services offered by things. Unfortunately, collecting and preparing such data is extremely time-consuming and even impossible due to lack of Internet of Things publically available. However, Zheng et al. [156] shared a large-scale real world services data in their WS-DREAM project. WS-DREAM is a crawling engine that crawled publicly available WSDL file addresses from the Internet. It also collected non-functional attributes (e.g., QoS) of these services, which are observed by 339 distributed computers located in 30 different countries, from Planet-Lab. In our experiments, we used this dataset as our base dataset and performed some pre-processing activities.

Firstly, we traversed all 5,825 WSDL addresses offered from the dataset and retrieved WSDL documents of 3,693 live services. We then generated an ontology by exploiting the approach developed in our previous work [107]. This ontology was bootstrapped by analyzing WSDL files and short textual descriptions of services using Term Frequency/Inverse Document Frequency (TF-IDF) and web context generation.

4http://www.planet-lab.org
The concepts from this ontology were used to annotate each service, which in turn generated the semantic description for the service. Consequently, each service is represented by a set of semantic service descriptors. Secondly, we collected the corresponding rating scores of services from websites such as seekda Web service search engine\(^5\), WebServiceLists, and ProgrammableWeb. Due to different rating scales used in these websites, we normalized the rating scores between the range of 1 to 5. Table 9.1 shows the dataset statistics for the original dataset and the one after the pre-processing. Figure 9.2 depicts the distribution of services and users (i.e., 339 computers).

\(^5\)Its site webservices.seekda.com is unavailable as of 08/05/2014.
and the throughput matrix (see Figure 9.3). For each of our 3,693 Web services, we extracted its response time and the throughput from the dataset. The aggregated QoS value of a service $s$ can be calculated using:

$$U(s) = \sum_{i \in \mathcal{A}} w_i \cdot \text{Score}_i(s) \quad (9.18)$$

where:

- $\text{Score}_i(s)$ is a QoS attribute scoring function, which, given a value of a QoS attribute $i$ of the service $s$, returns a score (a positive integer value). $\mathcal{A}$ is the set of QoS attributes.
- $w_i$ is the weight assigned to the QoS attribute $i$, and
- $w_i \in [0, 1]$ and $\sum_{i \in \mathcal{A}} w_i = 1$.

A scoring function is provided for each QoS attribute (in our case, response time and throughput) that calculates the score of the attribute for a particular service and scales the score to the interval [1..5]. A higher score value indicates a better quality of the service. However, we draw attention that some of the QoS attributes are negative (e.g., response time), i.e., the higher the value is, the lower the quality is. While others are positive (e.g., throughput), i.e., the higher the value is, the higher the quality is. Therefore the attribute scores should be scaled differently.

The rating scores from above aggregated QoS values were used for services for which we could not find a rating score from the Web sites. After the processing, we obtained the new dataset that was devoted to our experimental studies. The second half of Table 9.1 shows the statistical information of the new dataset. It should be noted that the users in this new dataset were eventually a combination of the real users
(who gave rating scores on the Web sites) and the computers used in the WS-DREAM project (that collected QoS values).

![Graphs showing response time and throughput](image)

**Figure 9.3:** QoS data collection in the WS-DREAM dataset (a) response time and (b) throughput

### 9.5.2 Metrics

We used the micro-F1 and macro-F1 as the evaluation measures in our experiments. The F1 measure is the harmonic mean of Precision ($P$) and Recall ($R$), which can be calculated as: $F_1 = \frac{2 \times P \times R}{P + R}$. The Micro-F1 is defined as:

$$Micro-F1 = \frac{2 \sum_{j=1}^{c} \sum_{i=1}^{n} \hat{y}_i^j y_i^j}{\sum_{j=1}^{c} \sum_{i=1}^{n} \hat{y}_i^j + \sum_{j=1}^{c} \sum_{i=1}^{n} y_i^j} \quad (9.19)$$

where $n$ is the number of test data, $y_i$ is the true label vector of the $i$-th sample, $y_i^j = 1$ if the instance belongs to category $j$, $-1$ otherwise. $\hat{y}_i$ is the predicted label vector. The micro-F1 measure weights equally on all samples, thus favoring the performance on common category labels. Macro-F1 is calculated as mean arithmetical value for F1 on each label. It measures weights equally on all the category labels regardless of how many samples belong to it, thus favoring the performance on rare category labels. Macro-F1 can be calculated using:
Chapter 9. Unified Collaborative Service Recommendation Combined with Content-based Features

\[ Macro \ F1 = \frac{2 \sum_{i=1}^{n} \hat{y}_i y_i^c}{n^2 |c|(\sum_{i=1}^{n} \hat{y}_i + \sum_{i=1}^{n} y_i^c)} \]  

(9.20)

To achieve more accurate evaluation, we considered the ranking position in the recommendation results and adopted the normalized Discounted Cumulative Gain (nDCG) as a metric, which is the normalized position-discounted precision score. It gives more credit to top-ranked Web services and is defined as below:

\[ DCG_x = \sum_{i=1}^{x} \frac{2^{rel_i} - 1}{\log(1 + p_i)} \]  

(9.21)

where \( p_i \) is the ranking position of \( s_i \) in top@\( x \) Web services, and \( rel_i \) is the relatedness score of \( s_i \) at position \( p_i \). The normalized DCG can be calculated using:

\[ nDCG_x = DCG_x / IDCG \]  

(9.22)

where \( IDCG \) is the maximum possible DCG till ranking position \( p_i \) of the sorted result list. The value range of nDCG is \([0, 1]\).

9.5.3 Performance Comparison

This experiment studies the recommendation performance of our proposed hybrid recommendation approach (HR) with the pure collaborative filtering methods (CF) and pure content-based recommendation (CBR). The recommendation accuracy was evaluated by examining the Top-\( N \) rankings for all users. Top-\( N \) recommendation is to recommend the \( N \) top-ranked items that will be of interest to a given user. For collaborative filtering methods, Top-\( N \) recommendation techniques analyze the user-item matrix to discover relations between different users or items, which are used to compute the recommendations. In our experiments, we considered three collaborative filtering methods [116]:

\[ Macro - F1 = \frac{2 \sum_{i=1}^{n} \hat{y}_i y_i^c}{n^2 |c|(\sum_{i=1}^{n} \hat{y}_i + \sum_{i=1}^{n} y_i^c)} \]  

(9.20)
• User-based Collaborative Filtering (UCF): This method first calculates the $k$ most similar users for a given user based on the Pearson correlation. The corresponding rows of the $k$ similar users in the user-item matrix are aggregated to identify a set of service items, which have been invoked by the group of users together with their ratings. With the identified service items, UCF then recommends the top-$N$ highly-rated service items that the target user has not invoked.

• Item-based Collaborative Filtering (ICF): Unlike UCF, this method first discovers $k$ most similar service items for each service item based on similarities. It then identifies the set as candidates of the recommended services by taking the union of the $k$ most similar items and removing each of the items in the set which the user has already invoked. The resulting set of the service items will be sorted in decreasing order based on the similarities and ICF then recommends the top-$N$ service items to a given user.

• Latent Factor Model (LFM): LFM [63] exploits matrix factorization, a main method in model-based CF recommender systems, based on singular value decomposition (SVD) with regularization.

In the experiment, the user-service interaction matrix (see Section 9.5.1) was randomly divided into the evaluation data matrix, $R_e$, and the training data matrix, $R_t$, by masking $p\%$ (also called markoff ratio) actual values of historical service invocation for each user. Figure 9.4 shows an illustrative example in dividing a simple user-service matrix (5 users and 4 service items). In the example, some user-service historical interactions are masked randomly as the evaluation data (shaded boxes), the rest data is used for learning the model parameters. After each model is learned, we used the model parameters to find $\forall s, Pr(s|u)$ for all users. The Web services in the testing dataset were ranked based on their $(Pr(s|u))$. 
Figure 9.4: Illustration of experiment implementation: we divided rating matrix $\mathcal{R}$ into the training matrix $\mathcal{R}_t$, in which partial rating scores are masked and the rest ratings are used to train our model, and evaluation matrix $\mathcal{R}_e$, in which the masked rating value can be predicted using our validated model.

The matrix (see Section 9.5.1) was randomly divided into the training data matrix, $\mathcal{R}_t$, and the evaluation data matrix, $\mathcal{R}_e$, by masking $p\%$ (also called markoff ratio) of actual values. Figure 9.4 shows an example in dividing a simple user-item matrix (5 users and 4 service items). In the example, 10% values are masked randomly as the evaluation data (shaded boxes).

The recommendation performance was evaluated by examining the quality of top $N$ rankings of services ($N = 1, 5, 10$). Specifically, after each model is learned, we used the model parameters to find $\forall s, Pr(s|u)$ for all users. The services in the testing dataset were ranked based on their $(Pr(s|u))$. In the experiment, we set the size of latent users’ preferences $\mathcal{N}_z$ is 50. We compared the recommended services with four other methods and our model (HR) by employing 10, 20, 30, 40 percent sparsity of the training dataset respectively. Table 9.2 shows the results.
## Table 9.2: Performance Comparison with Other Approaches

<table>
<thead>
<tr>
<th>TrainRatio</th>
<th>10%</th>
<th>20%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N = 1</td>
<td>0.6133</td>
<td>0.5683</td>
</tr>
<tr>
<td>N = 5</td>
<td>0.5779</td>
<td>0.5602</td>
</tr>
<tr>
<td>N = 10</td>
<td>0.5845</td>
<td>0.5794</td>
</tr>
<tr>
<td></td>
<td>0.6092</td>
<td>0.5615</td>
</tr>
<tr>
<td>MicroF1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UCF</td>
<td>0.6784</td>
<td>0.6383</td>
</tr>
<tr>
<td>ICF</td>
<td>0.6023</td>
<td>0.5494</td>
</tr>
<tr>
<td>CBR</td>
<td>0.5672</td>
<td>0.5512</td>
</tr>
<tr>
<td>LFM</td>
<td>0.5822</td>
<td>0.5531</td>
</tr>
<tr>
<td>HR</td>
<td>0.6627</td>
<td>0.6221</td>
</tr>
<tr>
<td>MacroF1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UCF</td>
<td>0.6762</td>
<td>0.6325</td>
</tr>
<tr>
<td>ICF</td>
<td>0.6279</td>
<td>0.5986</td>
</tr>
<tr>
<td>CBR</td>
<td>0.5916</td>
<td>0.5578</td>
</tr>
<tr>
<td>LFM</td>
<td>0.5673</td>
<td>0.5422</td>
</tr>
<tr>
<td>HR</td>
<td>0.7511</td>
<td>0.7248</td>
</tr>
<tr>
<td>nDCG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UCF</td>
<td>0.7288</td>
<td>0.6767</td>
</tr>
<tr>
<td>ICF</td>
<td>0.7011</td>
<td>0.6352</td>
</tr>
<tr>
<td>CBR</td>
<td>0.6329</td>
<td>0.6196</td>
</tr>
<tr>
<td>LFM</td>
<td>0.7312</td>
<td>0.6895</td>
</tr>
<tr>
<td>HR</td>
<td>0.7511</td>
<td>0.7248</td>
</tr>
<tr>
<td>TrainRatio</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Top N</td>
<td>0.7153</td>
<td>0.6635</td>
</tr>
<tr>
<td>N = 1</td>
<td>0.6894</td>
<td>0.6501</td>
</tr>
<tr>
<td>N = 5</td>
<td>0.6205</td>
<td>0.6072</td>
</tr>
<tr>
<td>N = 10</td>
<td>0.7105</td>
<td>0.6739</td>
</tr>
<tr>
<td>MacroF1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UCF</td>
<td>0.7436</td>
<td>0.7222</td>
</tr>
<tr>
<td>ICF</td>
<td>0.7153</td>
<td>0.6635</td>
</tr>
<tr>
<td>CBR</td>
<td>0.6894</td>
<td>0.6501</td>
</tr>
<tr>
<td>LFM</td>
<td>0.7105</td>
<td>0.6739</td>
</tr>
<tr>
<td>HR</td>
<td>0.7436</td>
<td>0.7222</td>
</tr>
<tr>
<td>nDCG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UCF</td>
<td>0.7323</td>
<td>0.6663</td>
</tr>
<tr>
<td>ICF</td>
<td>0.7115</td>
<td>0.6520</td>
</tr>
<tr>
<td>CBR</td>
<td>0.6325</td>
<td>0.6164</td>
</tr>
<tr>
<td>LFM</td>
<td>0.7233</td>
<td>0.6944</td>
</tr>
<tr>
<td>HR</td>
<td>0.7625</td>
<td>0.7339</td>
</tr>
</tbody>
</table>
From Table 9.2 we can see that, the top $N$ ($N$=1, 5, 10) performance of our approach (HR) are consistently higher than both UCF and ICF collaborative filtering methods, content-based recommendation approach (CBR), and regularized SVD (singular value decomposition) based latent factor model (LFM). It is clear that our approach outperforms the other approaches and more relevant Web services can be recommended. It also can be observed from the table that with the increase of the sparsity of our testing dataset\(^6\), the recommendation performance of UCF, ICF, CBR and LFM decreases steadily, they can not handle the data sparsity issue very well. However, the performance of our proposed approach remains relatively stable and is not sensitive to the sparsity level changes. This is largely contributed by the two strategies dealing with data sparsity introduced in Section 9.4.

Another aim of this experiment is to evaluate our approach on handling the cold start problem, which refers to providing accurate prediction when some users only access few Web services or even have no access historical records at all. From the table, it is clear that our approach outperforms other methods on different markoff ratios, especially when the training ratios are small (e.g., 10%). We also notice that the performance of UCF, ICF and LFM degrades significantly when the training ratio is 10%. Although the CBR is not sensitive to the training ratio, its overall performance is poor.

### 9.5.4 Impact of Markoff Ratio and Latent Variables

This section reports our experimental studies on the impact of markoff ratio and latent variables to the performance of our proposed hybrid approach. As mentioned previously, we divided the whole matrix into training and evaluation matrices. In the first experiment, we studied the impact of the markoff ratios. We randomly masked $y\%$ ($y$

\(^6\)Note that the sparsity of $\mathcal{R}_e$ is $1 - \text{markoff ratio.}$
Figure 9.5: Recommendation performance with different markoff ratios for top 1, 5, 10 Web services
Chapter 9. Unified Collaborative Service Recommendation Combined with Content-based Features

Figure 9.6: Recommendation performance with different number of latent variables $N_z$ for top 1, 5, 10 Web services
Chapter 9. Unified Collaborative Service Recommendation Combined with Content-based Features

= 10, 30, 50, 70) of actual scores and the rest of the matrix is used as training dataset to infer model parameters. Our algorithm was then used to recover the information that has been masked. We applied cross-validation method to find the average precisions and recalls for top $N$ ($N = 1, 5, 10$) service recommendations. Figure 9.5 shows the result. From the figure, we can see that with the increase of the markoff ratio, the overall recommendation performance decreases. This is attributed to the fact that a higher markoff ratio means less data available for training the approaches, therefore worse recommendation performance. Interestingly, we notice that the recommendation performance increases quickly when the markoff ratio decreases from 100% until it reaches 70%. After that, the recommendation performance increases slowly when the markoff ratio decreases.

To study the impact of the latent variables of our model, we conducted similar experiments but masked 30% of actual scores of the matrix. We ran the algorithm using different number of latent variables $N_z$ in the model, ranging from 10 to 80 with an interval of 10, and calculated Micro-F1, Macro-F1, and nDCG. Figure 9.6 shows the results. We can see that our system can mostly achieve the best recommendation performance around 50 latent variables. This is the reason that we chose $N_z = 50$ in our other experiments.

To study the impact of the latent variables of our model, we conducted similar experiments but masked 30% of actual scores of the matrix. We ran the algorithm using different number of latent variables $N_z$ in the model, ranging from 10 to 80 with an interval of 10, and recorded average precisions and recalls. Figure 9.6 shows the result. From the figure we can see that our system can achieve the best recommendation performance with 50 hidden variables. This is the reason that we chose $N_z = 50$ in our other experiments.
9.6 Related Work

Service recommendation and selection has been a fundamental research issue since the dawn of service technologies. The available service search engines such as XMethods largely exploit keyword-based search techniques and are inadequate to match the functionalities of services. These search engines do not consider non-functional characteristics (QoS) of services. Furthermore, users normally have to know how to craft correct queries. The performance of service recommendation of these search engines are therefore limited. Over the past few years, service recommendation has been an active research area and many techniques have been proposed. These techniques can be classified into three categories: collaborative filtering (CF), content-based, and hybrid approaches. In the following discussions, we will focus on reviewing these techniques.

9.6.1 CF Methods for Service Recommendation

The collaborative filtering methods are widely used in recommender systems that recommend items (services in our context) based on the similarity of different users. A representative research effort in this area has been done by Zheng et al. [156, 154]. In their work, a service QoS prediction approach is proposed to predict missing QoS values based on the historical QoS information from similar services and users. QoS-based services selection supports optimized services selection by considering QoS attributes of services with similar functionalities, as well as the preferences from service users [150, 119, 155]. The quality of the recommendation from these approaches depends on the quality of available QoS information for services. Most QoS-based service selection approaches assume that the QoS information (e.g., availability of service) is pre-existing and readily accessible with guaranteed quality, which unfortunately is not true, as indicated by Zheng et al. in [156]. Service providers may not be
able to deliver the QoS they promised and some QoS properties (e.g., network latency, invocation failure-rate, etc.) are highly related to the locations and network conditions of the service users. This makes such kind of approaches impractical to be used in many applications. By combining the traditional user-based and item-based collaborative filtering methods, the approach proposed by Zheng et al. [156, 154] can overcome the problem and mostly importantly, does not require service invocations in order to obtain the values of user-dependent QoS properties.

The work by Chen et al. [22] presents RegionKNN, a collaborative filtering algorithm that is designed for large-scale service recommendation. This approach considers service users’ physical locations and proposes a region model by considering the QoS characteristics of services. A refined nearest-neighbor algorithm is then developed for QoS-based service recommendation. In [109], Shao et al. propose a collaborative filtering based approach for mining users’ similarities and predicting QoS values for services. In a very recent effort, Yu et al. [147] specifically tackle the data sparsity issue by proposing an algorithm based on regularized Matrix Factorization.

Unfortunately, as discussed in the beginning of this paper, CF-based approaches have several inherent limitations. Since such approaches rely on interactions of services performed by other users, newly-deployed services cannot be recommended (i.e., the cold-start problem). In addition, the recommended services may be completely different (in terms of functionality) from the ones interacted by a given user in the past. In our work, we extend the collaborative filtering methods by considering semantic content similarities of services used by similar users. Our approach can effectively address the limitations of these CF-based service recommender systems.
9.6.2 Content-based Methods for Service Recommendation

The content-based approaches recommend items similar to those that a user appreciates based on the item’s characteristics (e.g., functionalities). The cold-start issue can be successfully solved by content-based approaches. However, such approaches typically require end users to know what keywords to use for a specific kind of services, which can be difficult for end users.

The techniques on content-based recommendation can be classified into two categories: syntactic and semantic based approaches. Syntactic-based methods focus on string manipulation and thesauri approaches to correlate services discovery. Woogle \cite{31} is one of the very earliest work in this direction. In Woogle, Dong et al. propose a service discovery approach based on matching users’ requirements and the functionalities of services. In particular, with the help of the co-occurrence of the terms appearing in service inputs and outputs, names of operations and descriptions in services, the authors develop a set of similarity search primitives that algorithms can use to match services. In a recent effort, WS-Finder \cite{82}, Ma et al. improve existing service discovery techniques by employing the Earth Mover’s Distance (EMD) for many-to-many partial matching between contents of user queries and service attributes. A $k$-NN algorithm is then used to produce top-$k$ services for users. Blake and Nowlan \cite{15} develop a service recommender system by exploiting an enhanced syntactic approach to compare the content of services. The approach aggregates and analyzes service messages and recommends services to end users. An interesting part of the work is the concept of naming tendency that is used to link strings from end users (e.g., queries) to the strings used in the definition of services (e.g., operation name, input and output). In general, syntactic-based approaches have limitations to suggest high quality recommendations.

In contrast, semantic-based methods recommend services by exploiting the sema-
tic description of their functionalities using ontological descriptions [107]. OWL-S\textsuperscript{7} is the first major ontology definition language for describing the semantics of services. There have been also efforts in developing languages for RESTful services such as the Web Application Description Language (WADL)\textsuperscript{8} and SA-REST [66]. In [67], Lécué and Delteil exploit a semantic similarity measure in services selection and composition. However, most existing semantic-based approaches focus only on standard semantic reasoning (i.e., subsumption) when inferring semantic similarities. Similarities between other parts (e.g., preconditions and effects) are seldom considered. In [68], Lécué continues his work in this direction and develops a complete specification of semantic service description by considering different levels of service recommendations.

9.6.3 Hybrid Methods for Service Recommendation

By combining both collaborative filtering and content-based methods, hybrid approaches for service recommendation can incorporate the advantages of the both methods while eliminating the weaknesses found in each approach [9, 116]. Hybrid approaches have improved prediction performance and overcome the cold start and sparsity problems of CF methods. Although hybrid recommendation approaches have been actively proposed in other areas such as e-Commerce [116], there is very limited work in the literature exploiting hybrid methods for service recommendation. The work presented in [68] is the only effort we are aware in this direction. Unfortunately, the paper largely focuses on proposing an approach to recommend services based on semantic content similarities of services. It remains unclear on how the collaborative filtering and content-based methods are integrated.

Our work presents a hybrid approach for better service recommendation by systematically combining both methods together. In particular, we propose a three-way

\textsuperscript{7}http://www.w3.org/Submission/OWL-S
\textsuperscript{8}http://www.w3.org/Submission/wadl
aspect model that considers both QoS ratings and the semantic content of services. User preferences are modeled as a set of latent variables in the aspect model [96], which can be statistically estimated using the expectation maximization (EM) method. To avoid overfitting problems caused by data sparsity, we further propose two strategies. The first one is to pre-process data matrix using a data smoothing technique and the second one is a modified aspect model that captures relationship between users and semantic content descriptors of services. To the best of our knowledge, our work is one of the first that combines collaborative filtering and content-based approach for service recommendation.

9.7 Conclusion

Things recommendation is a fundamental issue in Internet of Things since it assimilating more complicated and heterogeneous relationships in terms of people and things. Services offered by things can be exposed as services on the Internet for end user retrieval and invocation, recommending appropriate services in such scenario facing many unique challenges, as well as new perspectives towards the solutions. Existing service discovery and recommendation approaches focus on either the perishing registries, or keyword-dominant, QoS-based service search engines. Such approaches possess many limitations such as poor recommendation performance and heavily relying on the input from users (e.g., preparing correct queries).

In this chapter, we have proposed a novel hybrid approach for effective service recommendation. Our approach exploits a three-way aspect model that systematically combines classic collaborative filtering and content-based recommendation. The proposed hybrid approach simultaneously considers the similarities of user ratings and semantic content of services. Our approach is validated by conducting extensive experimental studies using 3,693 real-world services publicly available from the Internet.
The experimental results show that our approach outperforms the conventional collaborative and content-based methods in terms of recommendation performance.
Chapter 10

Conclusion and Future Work

In this dissertation, we analyze three kinds of relationships existing in Internet of Things, namely thing-to-thing, thing-to-attributes and thing-to-people, and propose models and approaches to exploit and leverage the relationships for further mining tasks such as annotation, clustering, classification and recommendation.

10.1 Conclusion

1. We propose four principles in mining relationships in Internet of Things Network, which are summarized as follows:

- **Richer patterns hide in people-things activities.** With more and more physical things are joining the Internet, effectively understanding relationships between people and smart objects is undeniably becoming a crucial task. Human behaviors in interacting with things are not completely random. Instead, certain patterns mostly exhibit in such behaviors.

- **Information propagate across people network and thing network.** People already form complex social networks, which are maintained and evolve by agreement on common objectives or shared interests and values. Links among people are well explored, however, the new thought is how
to propagate information across people and things in a holistic merged network containing people and things simultaneously.

- **Mining tasks by exploring intrinsic features of things.** Physical things are real entities in Internet of Things, and still carry some *intrinsic* unique features different from digital resources in Internet of Things Network. Studying these features is a new hot topic in research and applications.

- **Social-aware exploration of Internet of Things Network.** Things in reality have more close bonds with people comparing with digital resources. It means mining relationships in Internet of Things Network needs to pay more attention on human sociality, since human are social being. *People bestow richer social attributes to things.*

2. We have identified and extracted **three relationships** existing in Internet of Things Network:

   - **Thing-to-Thing Relationship (T2T-R).** The relationship indicates similarity between things and lays foundations for many important mining tasks and applications, such as things annotation and clustering.

   - **Thing-to-Attribute Relationship (T2A-R).** One or multiple attributes can be attached to things, like location information of things, temporal information and status etc. In this thesis, we focus on three attributes, which are thing’s availability, thing’s semantic labels and thing’s social tags.

   - **Thing-to-People Relationship (T2P-R).** This refers to a high level relationship indicating the people’s interest or preference on certain things. We consider this relationship as the recommendation relationship extended to Internet of Things scenario.

3. We have studied **different mining tasks** by exploring these **three relationships**
stemming from four principles, which includes correlation prediction (Chapter 2), classification (Chapter 5 and Chapter 6), annotation (Chapter 2), clustering (Chapter 2), recommendation (Chapter 8 and Chapter 9), and status prediction (Chapter 7). These mining tasks face unique challenges in Internet of Things Network, such as correlation prediction problem.

4. We have proposed a number of models and approaches to solve the above mining tasks, which include learning correlational network of ubiquitous things based on fusing contextual information of things usage events (T-DisCor/T-DisCor+ in Chapter 2) and things annotation framework (Chapter 3). Things classification via exploiting things attributes, i.e., semantic labels (Chapter 5) and social tags (Chapter 6), dynamic prediction of things availability (Chapter 7). The third relationship between people and thing which is interpreted by things recommendation, and it is accomplished via the propagation model (Chapter 8) and the probabilistic generative model (Chapter 9).

10.2 Future Directions

Internet of Things is still a young and promising research field. There are many challenging and unexplored research directions. Here we discuss some of them.

- Big Data Mining in IoT

More and more things are connecting to the Internet, Cisco Internet Business Solutions Group (IBSG) predicts that around 25 billion devices will be connected to the Internet by 2015, and 50 billion by 2020 \(^1\). Given the potentially vast amount of data stream from IoT systems, an increasingly important feature of

\(^1\)http://www.cio.com/article/745740/The_Internet_of_Things_Top_5_Threats_to_IoT_Devices
these systems will call more powerful and efficient approaches and algorithms to fuse, interpret, augment and present information from smart objects.

- **Self-* Internet of Things**

  In the emerging Internet of Things era, physical things need to discover each other as well as the resources provided by other things in the surroundings. The challenge here is that there is no fixed infrastructure to manage resource publication, discovery and communication. Each device should be capable of announcing its presence and the resources it provides without requiring a fixed infrastructure and also can discover other resources. This is unlike the traditional distributed systems where resource publication, discovery and communication are managed by a dedicated server. The server coordinates all the interactions between devices and is assumed to be in an always connected state. Self-* IoT aims at realizing plug-n-play, context-aware and autonomous Internet of things that will be self-discovering/aware, self-configuration, self-organizing, self-optimization and self-protection without (or with minimum) human intervention.

  - Self-discovering/aware: new things can be discovered and registered automatically,
  
  - Self-organizing: according to the changes in the dynamic environment (e.g., location change, connectivity change etc), deployed components can be organized in the run-time to keep performance,
  
  - self-configuration: things need to connect to the Internet to offer services have to announce and describe these services automatically in machine-understandable ways so that end users can find and utilize them. So that, end user systems do not need configure or program the things.
– self-optimization: things can perform automatic monitoring and control of resources to ensure the optimal functions or services offered with respect to the defined requirements.

– self-protection: things should be able to conduct proactive identification and protection from arbitrary attacks and can be recovered from damage.

**Things Search**

Effectively and efficiently find the stateful things and most appropriate things faces many unique challenges in IoT. One of major challenge is that physical resource always have location compared with conceptual resources. They can not be replicated or cached like conceptual resources. Plus, physical things have high mobility. As a result, building an effective and efficient discovery mechanism which can adapt and react very fast to the changes of physical things is very important. So far, not much work have been proposed on this topic.

**Highly Context-aware Internet of Things**

Context information of things is more enriched in Internet of Things, which includes location, status, time, role and activity etc. Physical things inherit some unique contextual features, for example, they could change their locations, status all the time, some physical things also have expire period, so catch the relationship between the physical things and their related context information is becoming more important as well as complicated. As a result, context-aware IoT needs to learn their surroundings and support more sophisticated and advanced query, aggregated and reasoning algorithms and techniques combining with semantics, intelligent and ontology techniques. For instance, we can induct the user’s future activity based on some data between he and relational things or implicit correlations between things (e.g., we have conducted some preliminary work on
things correlation discovery in Chapter 2), with which we can define rules and appropriate algorithms to find the most appropriate services and information for the specific user (e.g., we have developed some preliminary work on such things recommendation scenario in Part III).
BIBLIOGRAPHY


[59] Tatsuya Kameda, Yohsuke Ohtsubo, and Masanori Takezawa. Centrality in Sociocognitive Networks and Social Influence: An Illustration in


Appendix A

Curriculum Vitae

Lina YAO

PostDoc Research Fellow
School of Computer Science, University of Adelaide
North Terrace, Adelaide, SA 5005, Australia
Tel: +61 8 8313 1675
Email: lina.yao@adelaide.edu.au
Cyber Location: http://cs.adelaide.edu.au/~lina

EDUCATION

PhD in Computer Science, September 2010 - January 2014
The University of Adelaide, Australia
Thesis: Multidimensional Analysis of Heterogeneous Relationships in Internet of Things
Supervisors: A/Prof. Michael Sheng (University of Adelaide, Australia)
Prof. Anne H.H. Ngu (Texas State University, USA)

Master in Computer Science, July 2008 - July 2010
The University of Adelaide, Australia
Thesis: A Novel Approach to Automatic Verification of Web Services
Supervisor: A/Prof. Michael Sheng

Bachelor in Computer Science, September 1998 - July 2002
Shandong University, China
RESEARCH INTEREST

Web Mining, Internet of Things, Ubiquitous Computing, Web services, Software Engineering, Service Oriented Computing

RESEARCH EXPERIENCE

– Research Associate. School of Computer Science, University of Adelaide, 2014.1 - Present

HONORS AND AWARDS

– ACM Travel Grant for SIGIR 2014
– University of Adelaide Media Release for PhD work ¹, 2014
– Google Fellowship Nominee, 2013
– Google PhD Travel Award (AUS$2,500), 2012
– IBM PhD Travel Award for ICSOC 2011
– Highly Commended Research Poster Award, ARC EII PhD School 2010
– Adelaide Graduate Research Scholarship in Mathematical and Computer Science, July 2010 (for International students, only 3 scholarships are awarded in 2010)
– Ranked 1st in all Master of Computer Science coursework students in School of Computer Science, the University of Adelaide, July 2010
– Selective Travel Bursary from the ARC Research Network in Enterprise Information Infrastructure’s (EII) PhD Summer School, 2010
– Selective State-funded On-job Postgraduate studying from 2007
– Top 1% in On-job National Postgraduate Entrance Examination, Jinan, Shandong Province China, 2007

– Annual Employee Excellence Award, Department Telecommunications Network Maintenance, Jinan Branch, China Unicom, 2005, 2006
– Academic Merit Award, Shandong University, 1999, 2000

SUPERVISION


TUTORING

– COMP SCI 2005 (System Programming in C/C++), Semester 2, 2011 (107 students)
– COMP SCI 7036 (Software Engineering in Industry), Semester 1, 2011 (20 students)

PUBLICATION

Curriculum Vitae


4. Lina Yao, Quan Z. Sheng, Byron Gao, Anne H.H. Ngu and Xue Li. A Model for Correlation Discovery of Ubiquitous Things. The IEEE International Conference on Data Mining (ICDM 2013), December 7-10, Dallas, Texas, USA. (ERA A*)

5. Lina Yao, Quan Z. Sheng, Aviv Segev and Jian Yu. Recommending Web Services via Combining Collaborative Filtering with Content-based Features. The IEEE 20th International Conference on Web Services (ICWS 2013). June 27-July 2, 2013, Santa Clara Marriott, CA, USA (ERA A)


16. Lina Yao, Quan Z. Sheng. Particle Filtering based Availability Prediction for Web Services. The Ninth International Conference on Service Oriented Computing (ICSOC 2011), Paphos, Cyprus, December 5-8, 2011. (ERA A)

EXTERNAL SERVICES


2. PC member, The 13th IEEE International Conference on Ubiquitous Computing and Communications (IUCC 2014), Chengdu, China, December 19 -21, 2014.

3. PC member, The 2nd International Workshop on Future Services (FS 2014), Hilton Anchorage, Alaska, USA.


14. PC member, The First International Workshop on Sensor Data Processing and Integration. Beijing, China, August 21-24 2013

15. PC member, Theory and Practice in Modern Computing (TMPC 2013), Prague, Czech Republic, 2013


17. PC member, The Personal and Ubiquitous Computing (PUC) Special Issue on Advance in Context-aware Mobile Services (PUCSI-CAMS), 2013

19. PC member, The 1st International Conference on Data Analytics (DATA ANALYTICS 2012), Barcelona, Spain, September 23-28, 2012

20. PC member, Theory and Practice in Modern Computing conjunction with IADIS Multi Conference on Computer Science and Information Systems 2012 (TPMC 2012), Lisbon, Portugal, July 17-19, 2012


REVIEWER


2. Journal of Computer Science and Technology (JCST), 2014


4. Springer’s Handbook on Web Services 2012


INDUSTRY EXPERIENCE

Professional Telecommunications Engineer August 2003 - March 2008
Jinan Branch, China Unicom, Shandong, China
Assistant Telecommunication Network Engineer July 2002 - August 2003
Jinan Branch, China Unicom, Shandong, China

– Analyzing, troubleshooting, and maintaining multiple-level data network and equipment, includes the carrier class Internet: core nodes and back-
bone INTERNET of Jinan city and Shandong Province and several WAN circuits (ATM/DDN) of China Backbone through Shandong Province.
– Delivering and assisting VIP clients in network deployments and optimization.
– Optimizing and upgrading network data and equipment (LAN/MAN).

SELECTED PROJECTS

– Co-Supervised the massive and large scale optimization Jinan City MAN project in 2007, this project lasted about two months and included 4 core nodes (CISCO GSR 12816 Routers) and 19 convergence level nodes (CISCO 7609 Routers) etc. ()
– Led a project for optimizing and upgrading the LAN of Shandong Provin-
cial Hospital, one of the biggest hospitals in Shandong Province, in 2004
– Led an extensive testing of wide band subscriber behavior products of AL-
CATEL, HUAWEI, ZTE etc from Sep 2006 to Feb 2007
– Led the testing of QinQ on Huawei Switch & Redback SE800 Jan , 2007

REFEREES

ACADEMIC

A/Prof. Michael Sheng
School of Computer Science, The University of Adelaide
Adelaide, SA 5005, Australia
Tel: +61 8 8313 4197
Email: michael.sheng@adelaide.edu.au
Prof. David Suter
School of Computer Science, The University of Adelaide
Adelaide, SA 5005, Australia
Tel: +61 8 8313 3661
Email: david.suter@adelaide.edu.au

Prof. Anne H.H. Ngu
Department of Computer Science, Texas State University
San Marcos, TX 78666-4616, USA
Tel: +1 512 245 3409
Email: angu@txstate.edu

Ms. Chunli Li
Deputy General Manager
Telecommunication Network Maintenance Department, Jinan Branch, China Unicom
Tel: +86 15615319001
Addr: No.168, Jingshi Road, Jinan, China

Mr. Chengke Chen
Deputy Director
Department of Telecommunication Network Construction, Jinan Branch, China Unicom
Tel: +86 15615313297
Addr: No.168, Jingshi Road, Jinan, China