Abstract

An Intelligent Agent to Support Personalized Service

A lifestyle website is an electronic commerce website of providing daily necessities and services. But nowadays, most of them are lack of automatically notifying services and personalized learning mechanism. Therefore, users have to login the website and search suitable services in a large database. This is really a time-wasted and inefficient work.

This thesis proposed an extensible structure of the intelligent agent for the lifestyle website. We integrate three techniques to build this structure: (1) Data Cube structure, (2) Bit-Mapping technique, and (3) FP-Tree algorithm. With these techniques, the intelligent agent will not only analyze the user’ s shopping habits, but automatically notify users of suitable services before the habits happen. With this agent, the lifestyle website can mine more potential customers and let the user feel more convenient.

Keywords: E-Commerce, Partial Periodic Pattern, Intelligent Agent, Push Services

中文摘要

個人化服務支援之智慧型代理人

「生活網」是一種以提供日常必需品服務為主的電子商務網站。但是，現今大部分的「生活網」都缺乏所謂的主動服務通知跟個人化學習機制。因此，使用者必須登入網站，並且在眾多的資料裡，找出符合需求的服務，這實在是相當浪費時間，並且沒效率的服務方式。

本論文針對「生活網」，提出一個擴充性相當高的智慧型代理人架構。我們整合了三種技術來建造此架構：(1)資料立方體技術(Data Cube structure)，(2)位元比對技術(Bit-Mapping technique)，以及(3) FP-Tree 演算法。藉由這些技術，此智慧型代理人除了可以主動對使用者進行通知外，還可以進一步蒐集使用者的消費習慣，並在習慣發生前，預先通知使用者適當的服務。藉由此代理人，「生活網」便可以挖掘出更多的潛在顧客並且讓使用者感到更加的便利。

關鍵字：電子商務，部分週期樣版分析，智慧型代理人，主動服務

Contents

Tables

Figures

Chapter 1 Introduction

A lifestyle website is an electronic commerce website providing daily necessities and services. Nowadays, most of the famous lifestyle websites in Taiwan are lack of the “Push” mechanism. Therefore, the user must login the website, and search suitable services in such a large system. This is really a time-wasted and inefficient work.

What is the “Push” mechanism? That is, automatically provide the “right service” to the “right customer” in “right time.” Therefore, how to collect user’s shopping habits and analyze it correctly and efficiently are the key points to realize it.

This thesis proposed an intelligent agent for the lifestyle website. The agent will record the user habits, and automatically notify users of suitable services before the habits happen. For example, Laura has lunch in the restaurant near her home at about 12:10 every noon. After the agent records this habit, it will automatically login the lifestyle website and collect the suitable services for Laura before this habit happens. In order to realize it, the agent must retrieve the user’s basic shopping attributes (including *shopping time, location*, and *category*) and analyze these attributes in a correct way.

Personal agents are computer programs that can learn users' interests, preferences, and habits and give them proactive, personalized assistance with a computer application [1]. In other words, an agent must have the ability to retrieve basic shopping attributes and translate them to shopping habits in user's shopping transaction list. The shopping habits we called in this thesis, is a kind of *Full Periodic Patterns* and *Partial Periodic Patterns* (see chapter 2). Both of two patterns belong to Time-Series databases of data mining.

There are a lot of researches about data mining in Time-Series databases [6][7][8][9][10]. Among them, an interesting research [13] is similar to our approach. The author developed an Apriori-like algorithm to mine imperfect partial periodic pattern. But the Apriori pruning in mining partial periodicity may not be as effective as in mining association rules [11]. We use a more effective approach, FP-Tree [4], to improve the speed of mining. Furthermore, we also use FP-Tree to mine the relations between shopping behaviors to improve the accuracy of predicting habits.

The remaining of the thesis is organized as follows. In chapter 2, we will describe the background of our techniques. In chapter 3, we introduce the structure of our system. We list two kinds of habits in chapter 4 and analyze them in chapter 5 and 6 respectively. And in order to prove our research, we build up a tool to simulate it in chapter 7. In chapter 8, we have a discussion for our ideas. Finally, we conclude our study and introduce the future works in chapter 9.

Chapter 2 Background

2.1 A Multidimensional Data Model – Data Cube Structure [5][13]

A data cube allows data to be modeled and viewed in multiple dimensions. It is defined by dimensions and facts.

In general terms, **dimensions** are the perspectives or entities with respect to which an organization wants to keep records. For example [5], *AlEectronics* may create a *sales* data warehouse in order to keep records of the store's sales with respect to the dimensions *item, time, branch, and location.* These dimensions allow the store to keep track of things like monthly sales of items, and the branches and locations at which the items were sold. Each dimension may have a table associated with it, called a *dimension table*, which further describes the dimension. For example, a dimension table for *item* may contain the attributes *item_name, brand*, and *type*. Dimension tables can be specified by users or experts, or automatically generated and adjusted based on data distributions.

A multidimensional data model is typically organized around the central theme, like sales. The scheme is represented by a fact table. Facts are numerical measures. Think of them as the quantities by which we want to analyze relationships between dimensions.

Now, suppose that we would like to view the sales data with a third dimension. For instance, suppose we would like to view the data according to *time*, *item*, as well as *location* for the cities Chicago, New York, Toronto, and Vancouver. These 3-D data are shown in **Figure 1**[5]. The 3-D data of **Figure 1** are represented as a series of 2-D tables. Conceptually, we may also represent the same data in the form of a 3-D data cube, as in **Figure 2**.

	location = "Chicago"				location = "New York"				location = "Toronto"				location = "Vancouver"			
	item(Type)				item(Type)				item(Type)				item(Type)			
	home				home				home				home			
Time(quarter)	ent.	com.	pho.	sec.	ent.	com.	pho.	sec.	ent.	com.	pho.	sec.	ent.	com.	pho.	sec.
Q1	854	882	89	623	1087	968	38	872	818	746	43	591	605	825	14	400
Q2	943	890	64	698	1130	1024	41	925	894	769	52	682	680	952	31	512
Q3	1032	924	59	789	1034	1048	45	1002	940	795	58	728	812	1023	30	501
Q4	1129	992	63	870	1142	1091	54	984	978	864	58	784	927	1038	38	580

Figure 1: A 3-D view of sales data for *AllElectronics*, according to the dimensions *time, item,* and *location*. The measure displayed is *dollars_sold* (in thousands)

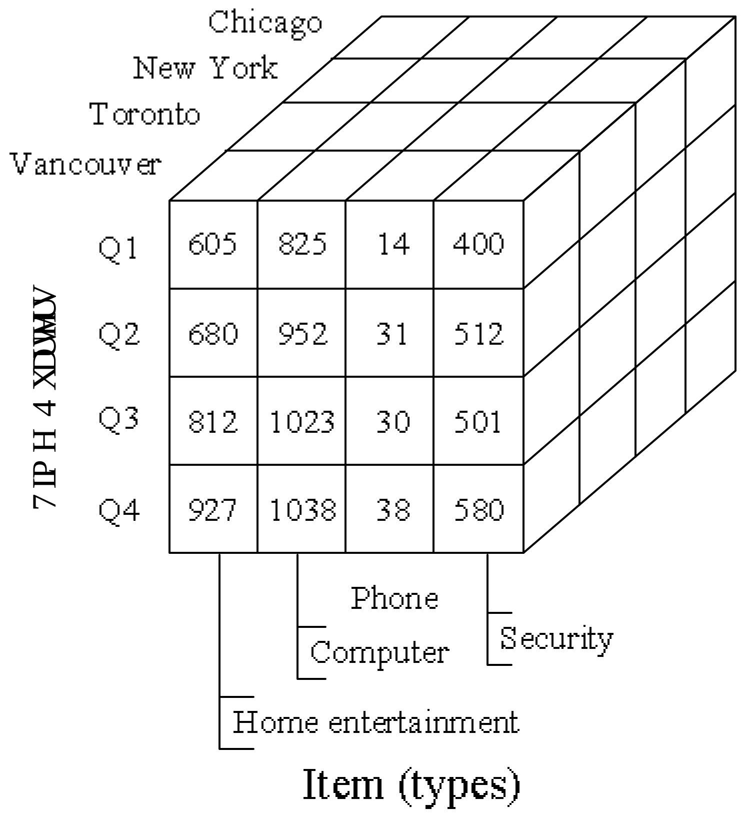


Figure 2: A 3-D cube representation of the data in Figure1

Suppose that we would now like to view our sales data with an additional fourth dimension, such as *supplier*. Viewing things in 4-D becomes tricky. However, we can think of a 4-D cube as being a series of 3-D cubes, as shown in **Figure 3**.

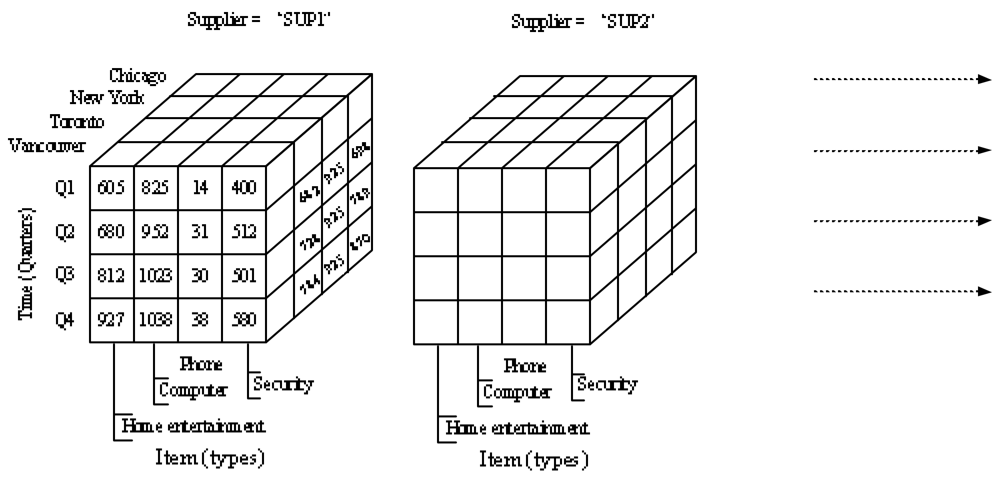


Figure 3: A 4-D data cube representation of sales data

In this thesis, we use the 4-D data cube structure to mine our habits. We will discuss it later.

2.2 Mining Association Rules in Large Database [5]

Association rule mining finds interesting association or correlation relationships among a large set of data items. With large amounts of data continuously being collected and stored, many industries are becoming interested in mining association rules from their databases. The discovery of interesting association relationship among huge amounts of business transaction records can help in many business decision making processes, such as catalog design, cross-marketing, and loss-leader analysis.

A typical example of association rule mining is ***market basket analysis***. This process analyzes customer's buying habits by mining association between the different items place in their "shopping baskets". The discovery of such associations can help retailers develop marketing strategies by gaining insight into which items are frequently purchased together by customers. For instance, if customers are buying milk, how likely are they also to buy bread on the same trip to the supermarket? Such information can lead to increased sales by helping retailers do selective marketing and plan their shelf space. For example, placing milk and bread within close proximity may further encourage the sale of these items together within single visits to the store.

2.3 The Apriori Algorithm: Finding Frequent Itemsets Using Candidate Generation[5][13]

Apriori is an influential algorithm for mining frequent itemsets for Boolean association rules. The name of the algorithm is based on the fact that the algorithm uses *priori knowledge* of frequent itemset properties. Apriori employs an iterative approach known as a *level-wise* search, where k-itemsets are used to explore $(k+1)$-itemsets. First, the set of frequent 1-itemsets is found. This set is denoted L_1. L_1 is used to find L_2, the set of frequent 2-itemsets, which is used to find L_3, and so on, until no more frequent k-itemsets can be found. The finding of each L_k requires one full scan of the database.

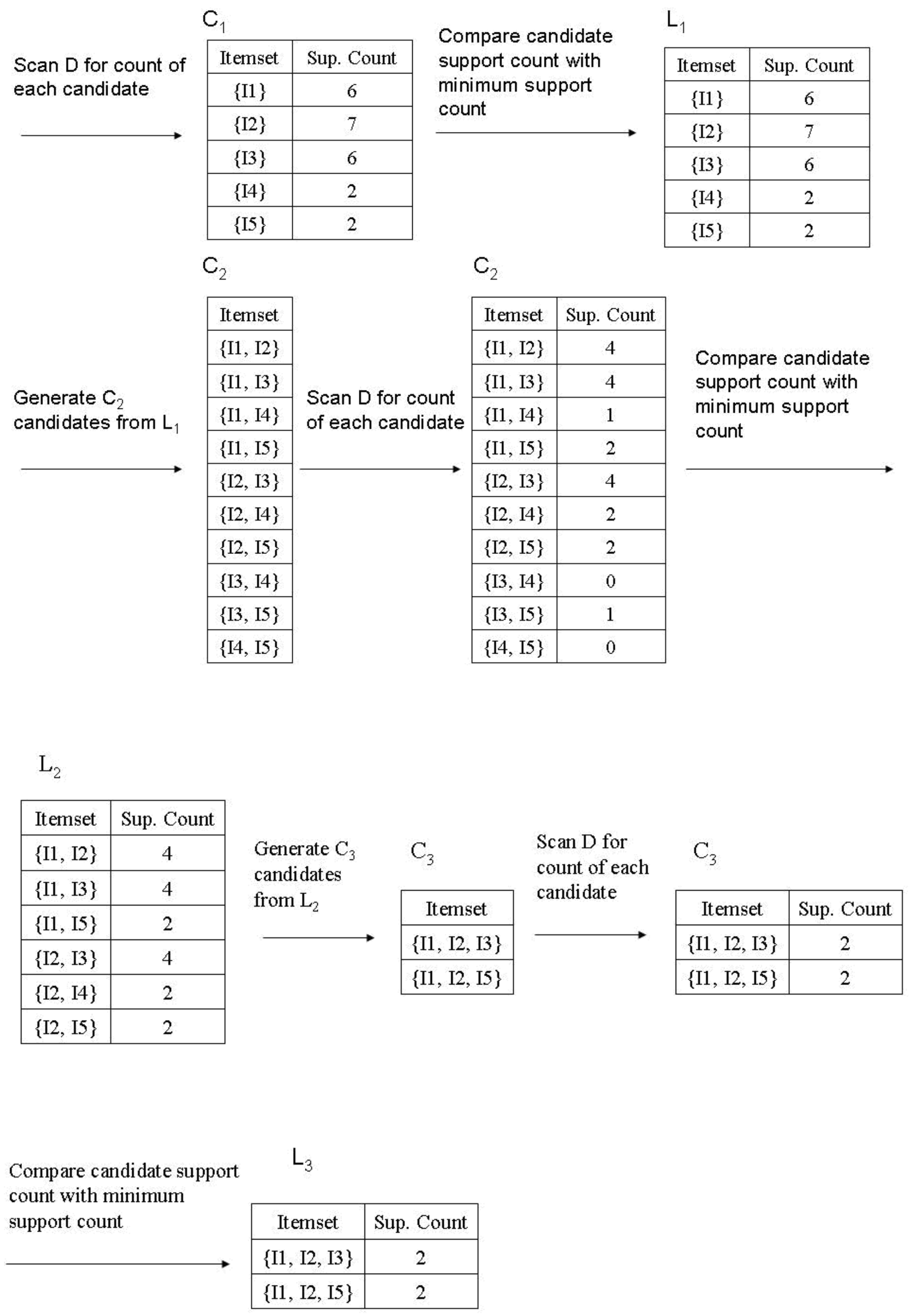
To improve the efficiency of the *level-wise* generation of frequent itemsets, an important property called the **Apriori property**, presented below, is used to reduce the search space.

Apriori property: All nonempty subsets of a frequent itemset must also be frequent. By definition, if an itemset I does not satisfy the minimum support threshold *(min_sup)*, then I is not frequent, that is, $P(I) < min\_sup$, where $P(I)$ is the probability of the itemset I happened. If an item A is added to the itemset I, then $A \cup I$ is not frequent either, that is, $P(A \cup I) < min\_sup$.

To understand the **Apriori property** used in the algorithm, we present an example from [5], illustrated by **Figure 4** and **Table 1**. **Table 1** shows a concrete example of Apriori, based on the *AllElectronics* transaction database, *D*. There are nine transactions in the database, that is, $|D| = 9$.

Table 1: Transactional data for an *AllElectronics* branch

TID	List of item_IDs
T100	I1, I2, I5
T200	I2, I4
T300	I2, I3
T400	I1, I2, I4
T500	I1, I3
T600	I2, I3
T700	I1, I3
T800	I1, I2, I3, I5
T900	I1, I2, I3



Scan D for count of each candidate $\longrightarrow$

C_1

Itemset	Sup. Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Compare candidate support count with minimum support count $\longrightarrow$

L_1

Itemset	Sup. Count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Generate C_2 candidates from L_1 $\longrightarrow$

C_2

Itemset
{I1, I2}
{I1, I3}
{I1, I4}
{I1, I5}
{I2, I3}
{I2, I4}
{I2, I5}
{I3, I4}
{I3, I5}
{I4, I5}

Scan D for count of each candidate $\longrightarrow$

C_2

Itemset	Sup. Count
{I1, I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

Compare candidate support count with minimum support count $\longrightarrow$

L_2

Itemset	Sup. Count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

Generate C_3 candidates from L_2 $\longrightarrow$

C_3

Itemset
{I1, I2, I3}
{I1, I2, I5}

Scan D for count of each candidate $\longrightarrow$

C_3

Itemset	Sup. Count
{I1, I2, I3}	2
{I1, I2, I5}	2

Compare candidate support count with minimum support count $\longrightarrow$

L_3

Itemset	Sup. Count
{I1, I2, I3}	2
{I1, I2, I5}	2

Figure 4: Generation of candidate itemsets and frequent itemsets, where the minimum support count is 2

1. In the first iteration of the algorithm, each item is a member of the set of candidate 1-itemsets, C_1. The algorithm simply scans all of the transactions in order to count the number of occurrences of each item.

2. Suppose that the minimum transaction support count required is 2 (*Example: min_sup = 2/9 = 22%*). The set of frequent 1-itemsets, L_1, can then be determined. It consists of candidate 1-itemsets satisfying minimum support.

3. To discover the set of frequent 2-itemsets, L_2, the algorithm uses $L_1 \rhd\lhd L_1$[1] to generate a candidate set of 2-itemsets, $C2$. $C2$ consists of $\binom{|L_1|}{2}$ 2-itemsets.

4. Next, the transactions in D are scanned and the support count of each candidate itemset in C_2 is accumulated, as shown in the middle table of the second row in **Figure 4**.

5. The set of frequent 2-itemsets, L_2, is then determined, consisting of those candidate 2-itemsets in C_2 having minimum support.

6. C_3 is the generation of the set of candidate 3-itemsets. Let $C_3 = L_2 \times L_2$

[1] $L_1 \rhd\lhd L_1$ is equivalent to $L_1 \times L_1$ since the definition of $L_k \times L_k$ requires the two joining itemsets to share $k - 1 = 0$ items.

= *{{I1, I2, I3}, {I1, I2, I5}, {I1, I3, I5}, {I2, I3, I4}, {I2, I3, I5}, {I2, I4, I5}}*. Based on the Apriori property that all subsets of a frequent itemset must also be frequent, we can determine that the four latter candidates cannot possibly be frequent. We there fore remove them from C_3, thereby saving the effort of unnecessarily obtaining their counts during the subsequent scan of D to determine L_3. Note that when given a candidate k-itemset, we only need to check if its (K-1)-subsets are frequent since the Apriori algorithm uses a level-wise search strategy.

7. The transactions in D are scanned in order to determine L_3, consisting of those candidate 3-itemsets in C_3 having minimum support.

8. The algorithm use $L_3 \bowtie L_3$ to generate a candidate set of 4-itemsets, C_4. Although the join results in *{{I1, I2, I3, I5}}*, this itemset is pruned since its subset *{{I2, I3, I5}}* is not frequent. Thus, $C_4 = \phi$, and the algorithm terminates, having found all of the frequent itemsets.

2.4 Frequent Pattern Tree: Finding Frequent Patterns without candidate generation[5][6]

In many cases, the Apriori candidate generate-and-test method reduces the size of candidate sets significantly and leads to good performance gain. However, it may suffer from two nontrivial costs as follows:

- ✓ **It may need to generate a huge number of candidate sets.** For example, if there are 10^4 frequent 1-itemsets, the Apriori algorithm will need to generate more than 10^7 candidate 2-itemsets and accumulate and test their occurrence frequencies. Moreover, to discover a frequent pattern of size 100, such as $\{a_1, \ldots, a_{100}\}$, it must generate more than $2^{100} \sim 10^{30}$ candidate in total.

- ✓ **It may need to repeatedly scan the database and check a large set of candidates by pattern matching.** This is especially the case for mining long patterns.

In order to solve these two drawbacks, we adapt the FP-Tree technique to mine the relationship between the shopping behaviors. We reexamine the mining of transaction database, *D*, of **Table 1** in chapter

2.3. We will use FP-Tree technique to find frequent patterns again.

The first scan of the database is the same as Apriori, which derives the set of frequent items (1-itemsets) and their support counts (frequencies). Let the minimum support count be 2. The set of frequent items is sorted in the order of descending support count. This resulting set of *list* is denoted *L*. Thus, we have *L = {I2:7, I1:6, I3:6, I4:2, I5:2}.*

As FP-Tree technique is then constructed as follows. First, create the root of the tree, labeled with "null". Scan database *D* a second time. The items in each transaction are processed in *L* order (i.e., sorted according to descending support count) and a branch is created for each transaction. For example, the scan of the first transaction, "T100: I1, I2, I5", which contains three items (I2, I1, I5) in *L* order, leads to the construction of the first branch of the tree with three nodes: {(I2:1), (I1:1), (I5:1)}, where I2 is linked as a child of the root, I1 is linked to I2, and I5 is linked to I1. The second transaction, T200, contains the items I2 and I4 in *L* order, which would result in a branch where I2 is linked to the root and I4 is linked to I2. However, this branch would share a common **prefix,** I2, with the existing path for T100. Therefore, we instead increment the count of the I2 node by 1, and create a new node, {I4:1}, which is linked as a child of {I2:2}. In general, when considering the branch to be added for a transaction, the count of each node along a common prefix is incremented by 1, and nodes for the items following the prefix are created and linked accordingly.

To facilitate tree traversal, an item header table is built so that each item points to its occurrences in the tree via a chain of **node-links**. The tree obtained after scanning all of the transactions is shown in **Figure 5** with the associated node-links. Therefore, the problem of mining frequent patterns in databases is transformed to that of mining the FP-Tree.

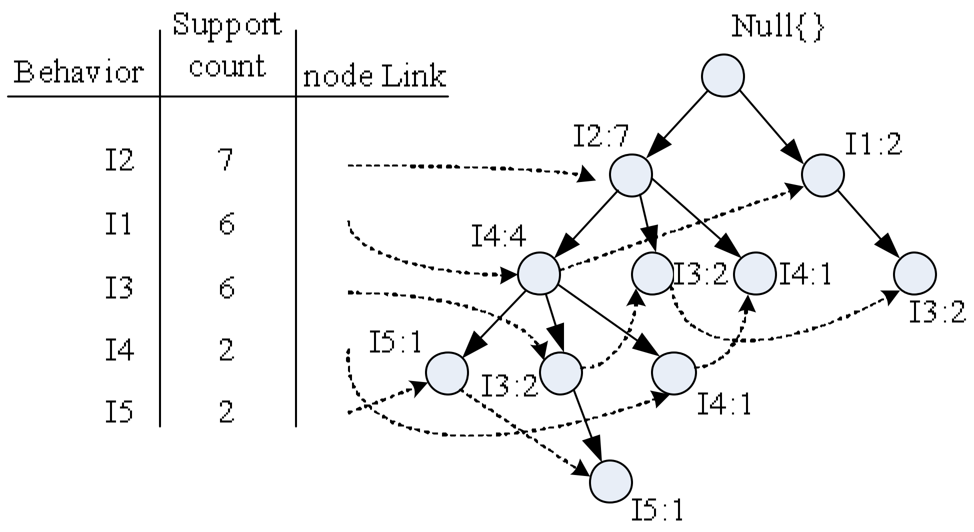


Figure 5: An FP-Tree that registers compressed, frequent pattern information

The mining of the FP-Tree proceeds as follows. Start from each frequent length-1 pattern (as an initial **suffix pattern**), construct its conditional pattern base (a “subdatabase”) which consists of the set of *prefix paths* in the FP-Tree co-occurring with the suffix pattern), then construct its (*conditional*) FP-Tree, and perform mining recursively on such a tree. The pattern growth is achieved by the concatenation of the suffix pattern with the frequent patterns generated from a conditional FP-Tree.

Mining of the FP-Tree is summarized in **Table 2** and detailed as follows. Let's first consider I5 which is the last item in *L*, rather than the first. The reasoning behind this will become apparent as we explain the FP-Tree mining process. I5 occurs in two branches of the FP-Tree of **Figure 5**. The paths formed by these branches are {(I2 I1 I5: 1)} and {(I2 I1 I3: 1)}. Therefore, considering I5 as a suffix, its corresponding two prefix paths are {(I2 I1: 1)} and {(I2 I1 I3: 1)}, which form its conditional pattern base. Its conditional FP-Tree contains only a single path, {(I2: I1: 2)}; I3 is not included because its support count of 1 is less than the minimum support count. The single path generates all the combinations of frequent patterns: I2 I5:2, I1 I5: 2, I2 I1 I5: 2.

For I4, its two prefix paths form the conditional pattern base, {(I2 I1: 1), (I2: 1)}, which generates a single-node conditional FP-Tree (I2: 2) and derives one frequent pattern, I2 I4: 2.

Similar to the above analysis, I3's conditional pattern base is {(I2 I1: 2), (I2: 2), (I1: 2)}. Its conditional FP-Tree has two branches, (I2: 4, I1: 2) and (I1: 2), which generates the set of patterns: {(I2 I3: 4), (I1 I3: 2), (I2 I1 I3: 2)}. Finally, I1's conditional pattern base is {(I2: 4)}, whose FP-Tree contains only node (I2: 4), which generates one frequent pattern, I2 I1: 4.

Table 2: Mining the FP-Tree by creating conditional (sub)pattern bases

Item	Conditional pattern base	Conditional FP-Tree	Frequent patterns generated
I5	{(I2 I1:1), (I2 I1 I3: 1)}	(I2: 2), (I1: 2)	I2 I5: 2, I1 I5: 2, I2 I1 I5:2
I4	{(I2 I1: 1), (I2: 1)}	(I2: 2)	I2 I4: 2
I3	{(I2 I1: 2), (I2: 2), (I1: 2)}	(I2: 4, I1: 2), (I1: 2)	I2 I3: 4, I1 I3: 4, I2 I1 I3: 2
I1	{(I2: 4)}	(I2: 4)	I2 I1: 4

At last, we get the same frequent pattern as Apriori algorithm. But the FP-Tree technique is more efficient than Apriori algorithm[4], so we adapt it to mine the related shopping behaviors.

2.5 **Mining Time-Series and Sequence Data[5]**

A time-series database consists of sequences of values or events changing with time. The values are typically measure at equal time intervals. Time-series databases are popular in many applications, such as studying daily fluctuations of a stock market, traces of a dynamic production process, scientific experiments, medical treatments, and so on. A time-series database is also a sequence database. However, a sequence database is any database that consists of sequences of ordered events, with or without concrete notions of time. For example, Web page traversal sequences are sequence data, but may not be time-series data.

In our thesis, we will discuss the **periodicity analysis.** Periodicity analysis is the mining of periodic patterns, that is, the search for recurring patterns in time-series databases. Periodicity analysis can be applied to many important areas. For example, seasons, tides, planet trajectories, daily power consumptions, daily traffic patterns, and weekly TV programs all present certain periodic patterns.

The problem of mining periodic patterns can be partitioned into three categories:

- ✓ **Mining full periodic patterns**, where every point in time contributes (precisely or approximately) to the cyclic behavior of the time series. For example, all of the days in the year approximately contribute to the season cycle of the year.

- ✓ **Mining partial periodic patterns,** which specify the periodic behavior of the time series at some but not all of the points in time. For example, Laura has lunch in the restaurant near her home at about 12:10 every noon, but her activities at other times do not have much regularity. Partial periodicity is a looser form of periodicity than full periodicity, and it also occurs more commonly in the real world.

- ✓ **Mining cyclic or periodic association rules,** which are rules that associate a set of events that occur periodically. An example of a periodic rule is "*Based on day-to-day transactions, if afternoon tea is well received between 3:00-5:00 pm, dinner will sell well between 7:00-9:00 pm on weekends.*"

In this thesis, mining the habits is similar to mine partial periodic patterns. After we mine the habits, we will automatically notify users of suitable services before the shopping habits happened. With this mechanism, users will not need to waste time searching the services they want, and the stores will have more chances to sale their goods.

2.6 The Intelligent Agent

In this thesis, the agent will not only record the user's shopping behaviors, but automatically notify users of suitable services. But, what are personal agents? Personal agents are computer programs that learn users' interests, preference, and habits and give them proactive, personalized assistance with a computer application [15].

What is the different between expert systems? Expert systems are computer programs designed to emulate the behavior of a human being who is expert at solving problems within a specialized area. Intelligent agents are computational entities capable of autonomously achieving goals by executing needed actions [16].

The intelligent agents have four common agent characteristics that have been identified in [17]:

- Situatedness: Agents receive sensory information from their surroundings and perform actions based on this information.
- Autonomy: Agents maintain their own internal state and are able to act without direct intervention from humans or other agents.
- Adaptivity: Agents are able to react to a changing environment. Agents with this characteristic can learn from experience and create goals to be achieved.
- Sociability: Agents have the ability to confer with other agents and/or

humans.

In our tool simulation, the agent we proposed have last three characteristics. We do not have a large lifestyle website to collect user shopping behaviors, so we lack of first characteristic. We must record user's shopping behaviors by user input them, and it's a critical problem we need to solve in the future.

There are many different types of agent in data mining. Lewis [18] categorizes three general types of intelligent agents—anticipatory agents, filtering agents, and semiautonomous agents.

- **Anticipatory agents** attempt to anticipate the intentions of a user. A rudimentary example of an anticipatory agent is Microsoft Word's animated paperclip. A more complex anticipatory agent might do our bidding on Ebay once we provide a product description, bidding criteria (e.g., only bid on items sold by someone who has sold previous items through Ebay), and a maximum offering price.

- **Filtering agents** are able to carry the categorization process one step further in that they can evaluate, prioritize, and delete information such as incoming e-mail messages. MAXIMS [19] and MAGI [20] are two systems offering these types of filtering capabilities.

- **Semiautonomous agents** deal with tasks requiring complex sequences of actions. The agent is provided with a goal to achieve by interacting with the user through a form-based interface. A typical goal is to have the agent search for an item, such as an airline ticket, which meets a set of constraints. Once found, the agent notifies the user via e-mail. The user is then responsible for making the actual ticket purchase.

There are still a lot of researches about intelligent agents. If you are interested in it, you can read more from [21][22][23].

Chapter 3 Structure of the System

Figure 6 shows the structure of the system we proposed. There are two ways to provide services for the users. The first way is that the agent receives the users' request "passively", and searches the suitable services from the service profile to notify him. The second way is that when the agent learns the user's habits, it will "actively" notify the user of suitable services before the habits happen. This thesis was focus on the personalized learning mechanism and we will not specify other functions in detail. We will discuss them in our future writings.

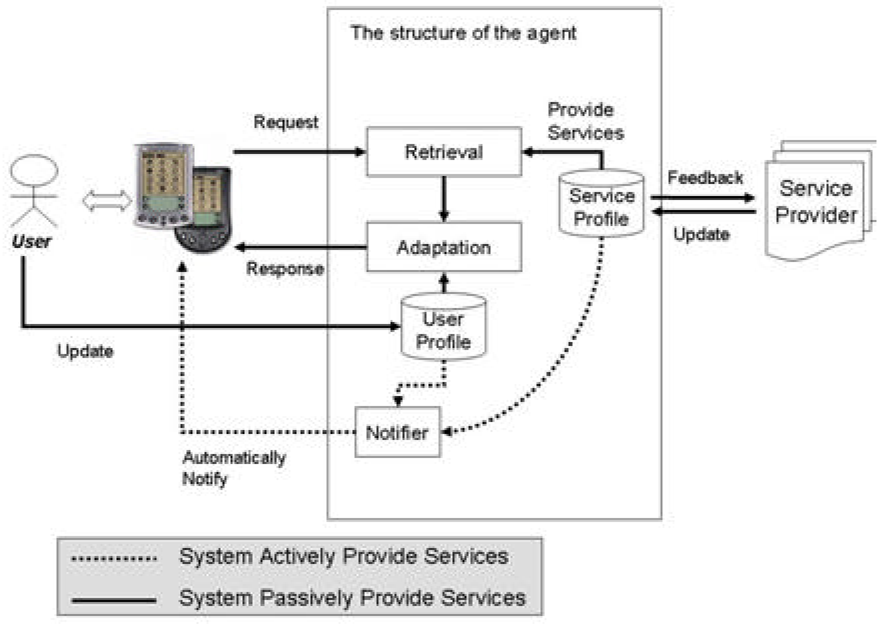


Figure 6: The structure of the system

The intelligent agent will divide a day into 24 segments (hours). The agent will automatically notify users of suitable services before the old segment pass. Services can be easily pushed to the right users through the agent. In order to realize the "push" ability, the agent must have two mechanisms as follows:

- **Automatic notification mechanism**

  Most of the lifestyle websites are lack of the automatic notification mechanism, so the services were in a passive situation to wait for customers finding. In our system, the agent will use RSS2.0 (Really Simple Syndication 2.0) [12] to notify users.

  There are three notification channels. They are "Advertisement", "User subscribed", and "Personalized" channels. The channel of "Advertisement" and "User subscribed" are not the key points in this thesis, so we will not discuss them here. The agent will use personalized learning mechanism to notify users of suitable services through the "Personalized" channel.

- **Personalized learning mechanism**

  The shopping habits we proposed, including **V**ertical **H**abits

(VH) and **H**orizontal **H**abits (HH). Vertical habits are the periodic shopping habits and horizontal habits are the relations between shopping behaviors. For example, Laura has lunch at 12:10 every Monday (VH), and she likes to buy drinks after lunch except Monday (HH).

After integrating vertical habits and horizontal habits, the agent will notify Laura of drinks' services after she finished lunch every Monday. The purpose of this approach is to reduce the deviation when the user had a temporary trip and improve the accuracy of predicting habits.

In order to realize this mechanism, we integrated three techniques to build it: (1) Data Cube structure [5][13], (2) Bit-Mapping technique, and (3) FP-Tree algorithm. It shows that data cube structure provides an efficient and effective structure for on-line analytical processing (OLAP) and on-line analytical mining [5]. As to FP-Tree algorithm, it mines frequent patterns without candidate generation. It is also more effective than Apriori algorithm [4].

We retrieve user's vertical habits based on data cube structure, and use FP-tree to retrieve horizontal habits.

Chapter 4 Vertical Habits (VH) and Horizontal Habits (HH)

Vertical habits are the periodic shopping habits. We denote $VH = (Day, B_i)$. B_i is the number of shopping **b**ehaviors. "*Day*" is the day when the shopping habits happened, where $Day = \{1, 2, \ldots, 7\}$.

Horizontal habits are the relations between shopping behaviors. We denoted $HH = (B_i, \{B_j\})$, where B_j is a set of shopping behaviors, i ? j, $i > 0$, $j \geqq 0$ ($j = 0$ means that there are not any related behaviors with B_i)

After integrating horizontal habits and vertical habits, we can call it **C**yclic **P**attern, $CP=(Day, \{BH\} \cup \{HH\})$. The cyclic patterns are also the shopping habits we called in this thesis.

Chapter 5 Mining Vertical Habits – Using Data Cube-based Structure

In this chapter, we discuss how the agent mining vertical habits with data cube-based structure after it record basic shopping attributes (location, category and time).

We construct two data cubes, *template cube* and *mining cube*. Template cube is used to record users' basic attributes and mining cube is used to mine vertical habits.

- **Template Cube**

Template cube is a 4-D data cube. It records the basic shopping attributes, including *shopping location*, *category of services*, *shopping segments,* and *shopping day*. The concept hierarchies for these attributes are as follows:

- ✓ *Shopping location:* shopping district→city
- ✓ *Category of services:* food, clothes, live, traffic, education, entertainment, and the others.

- ✓ *Shopping segments:* 24 hours

- ✓ *Shopping day:* day→week→month→year.

The number of the cell represents the number of **B**ehaviors (B_i), where $B_i = (Location, Service, segment, Day)$, $i = 1, 2, 3, \ldots, n$, $n > 0$, and *n* is the total number of the cells.

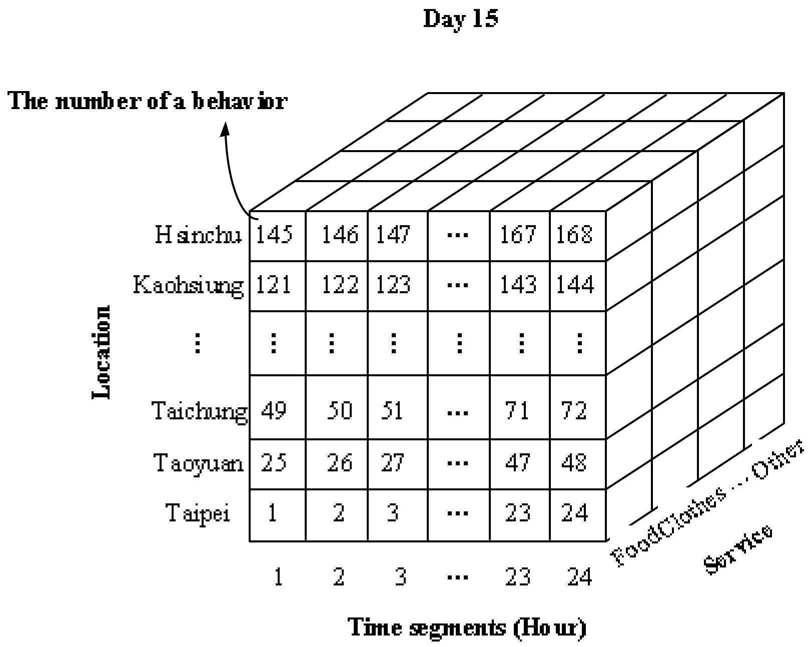


Figure 7: Template Cube

Figure 7 is the template cube based on shopping environment in Taiwan. For example, B_{145} = *(Hsinchu, Food, 1, 15)*. When the agent record B_{145}, it shows that the user *have a night snack* at *1:00* in *Hsinchu* in *Day15*. After retrieving the shopping information, we can mine vertical habits using mining cube.

➢ Mining Cube

Mining cube, as **Figure 8**, is used to mine vertical habits. We fold time dimension into two parts: *time-index* dimension and *period-index* dimension. Notice that each cell needs only a bit (existent, nonexistent). Each slice of the mining cube can be implemented as a bit-array, except the last slice, *period-index* = *All*, which contains the number of nonzero bits of all weeks slices and each cell is an integer.

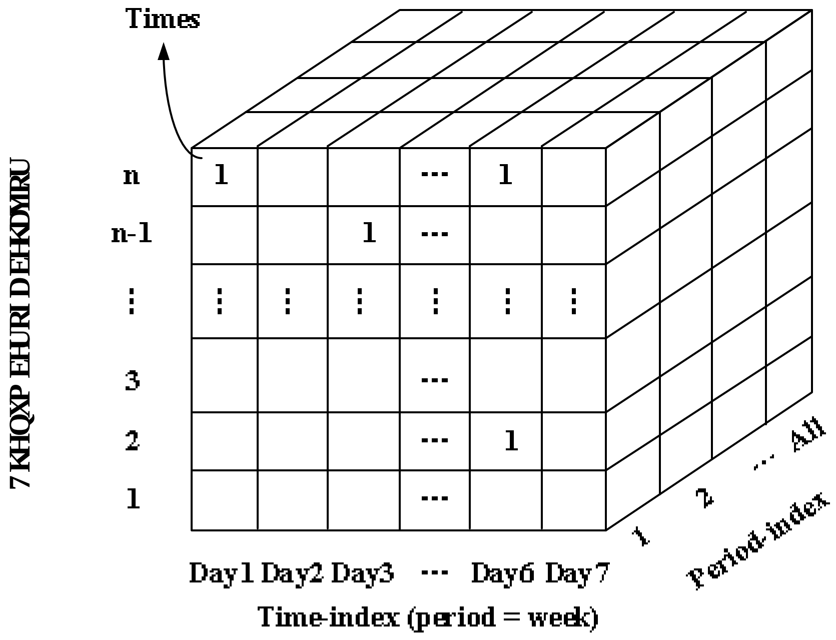


Figure 8: Mining Cube

Through mining cube, we can easily record the shopping behaviors. But in the real world, a habit will not be happened with 100%. So we must introduce the concept of *confidence* to tolerate misses. We denote *confidence* as a, *total number of period-index* as *ß*, and *minimum support* as $a \times ß$. The vertical habits must no less than minimum support.

For example, as **Figure 9**, if we assume $a = 50\%$ and $ß = 4$ (weeks), we can figure out that *minimum support* = 2 (50% × 4). Now, only *(Day1, B_n)*, *(Day3, B_{n-1})* and *(Day6, B_2)* can pass the threshold, and they are the vertical habits.

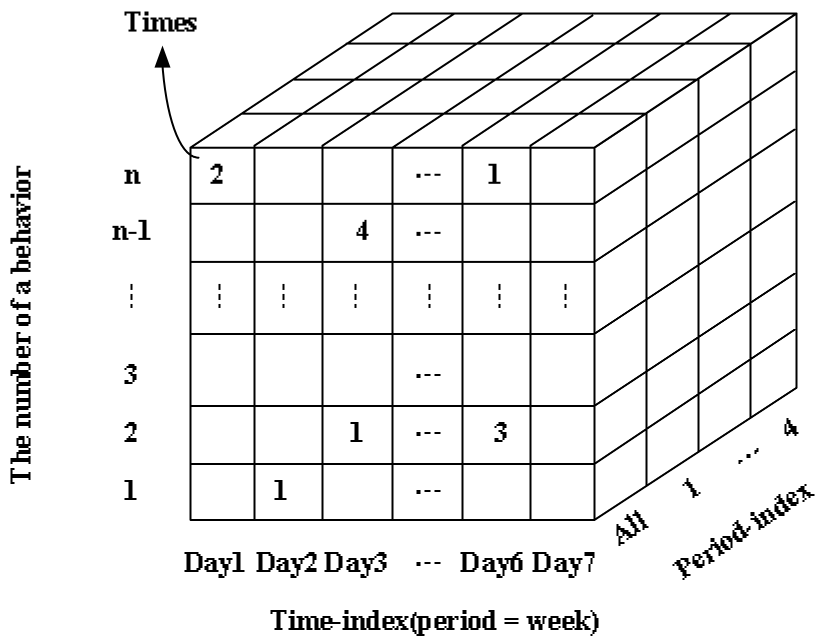


Figure 9: Mining cube where period-index = All

Chapter 6 Mining Horizontal Habits–Using FP-Tree Algorithm

After analyzing vertical habits, we almost know the user's habits. But if user has a temporary trip, it will induce the deviation. In order to solve this problem, we use FP-Tree to retrieve horizontal habits and improve the accuracy of predication.

We made a table for example, as **Table 3**.

Table 3: An example of vertical habits

Time Period	The number of a Behavior
Day1	B_1, B_2, B_5
Day2	B_2, B_4
Day3	B_2, B_3
Day4	B_1, B_2, B_4
Day5	B_1, B_3, B_5
Day6	B_1, B_2, B_6
Day7	B_1, B_2, B_3, B_5

We assumed that the minimum support is 3. First, we scan **Table 3** and derive **Table 4** after deleting the infrequent behaviors (less than 3). Through **Table 4**, we derive a list of frequent items, *{B_2: 6, B_1: 5, B_3: 3, B_5: 3},* and translate it to FP-Tree, as **Figure 10**.

Table 4: The frequent behaviors of vertical habits

Time Period	The number of a Behavior
Day1	B_2, B_1, B_5
Day2	B_2,
Day3	B_2, B_3
Day4	B_2, B_1
Day5	B_1, B_3, B_5
Day6	B_2, B_1
Day7	B_2, B_1, B_3, B_5

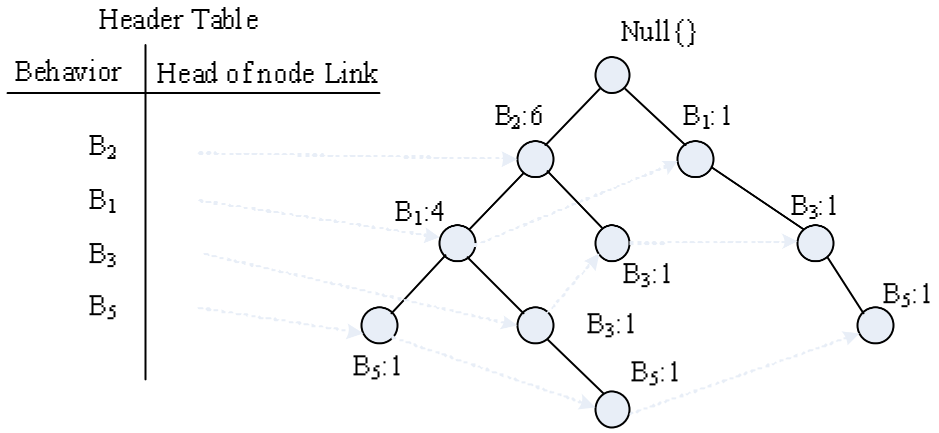


Figure 10: The FP-Tree in Table 4

The conditional pattern bases and the conditional FP-Trees generated are summarized in **Table 5**. Through **Table 5**, we found the horizontal habits, *HH = ($B_{1,}$ B_2)*. Notice that through template cube, we can know which behaviors happened earlier.

Table 5: Mining of all-patterns by creating conditional (sub) - pattern bases

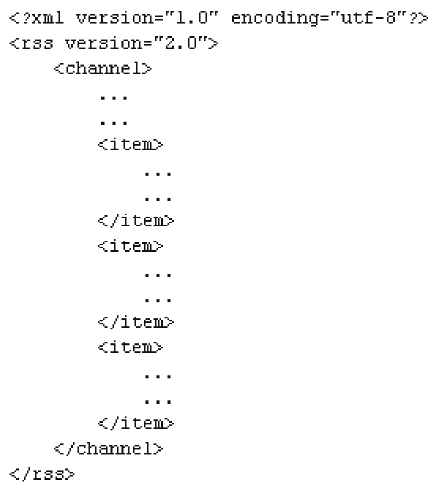
Behavior ID	Conditional Pattern Based	Conditional FP-Tree	Sequence Pattern
B_5	$\{(B_2 B_1{:}1), (B_2 B_1 B_3{:}1)\}$	Null	null
B_3	$\{(B_2 B_1{:}1), (B_2{:}1), (B_1{:}1)\}$	Null	null
B_1	$\{(B_2{:}4)\}$	$(B_2{:}4)$	$B_2 B_1{:}4$

After integrating vertical habits and horizontal habits, we modify the habits of "*Day 5*", translate to cyclic patterns, as **Table 6**, and it is the final habits we proposed.

Table 6: Cyclic Patterns

Cyclic Pattern(CP)
Day1, B_1, B_2, B_5 Day2, B_2, B_4
Day3, B_2, B_3
Day4, B_1, B_2, B_4
Day5, B_1, $\boldsymbol{B_2}$, B_3, B_5
Day6, B_1, B_2, B_6
Day7, B_1, B_2, B_3, B_5

Finally, after we retrieve the shopping behaviors (*CP*), we will automatically notify the user of suitable services. How to automatically notify a user? Because we know the user' shopping behavior hour and the day, the server will automatically generate the RSS Seed in the personalized channel. **Figure 11** shows a template structure of a RSS 2.0 seed. From the specification of RSS2.0, we can know the seed's information [14]. We can know what kind of services the seed belong by the ***category*** element and what time it broadcast by the ***pubDate*** element. With these two information, we can easily search the seed of right category service and right time satisfying the user's habits.



```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0">
    <channel>
        ...
        ...
        <item>
            ...
            ...
        </item>
        <item>
            ...
            ...
        </item>
        <item>
            ...
            ...
        </item>
    </channel>
</rss>
```

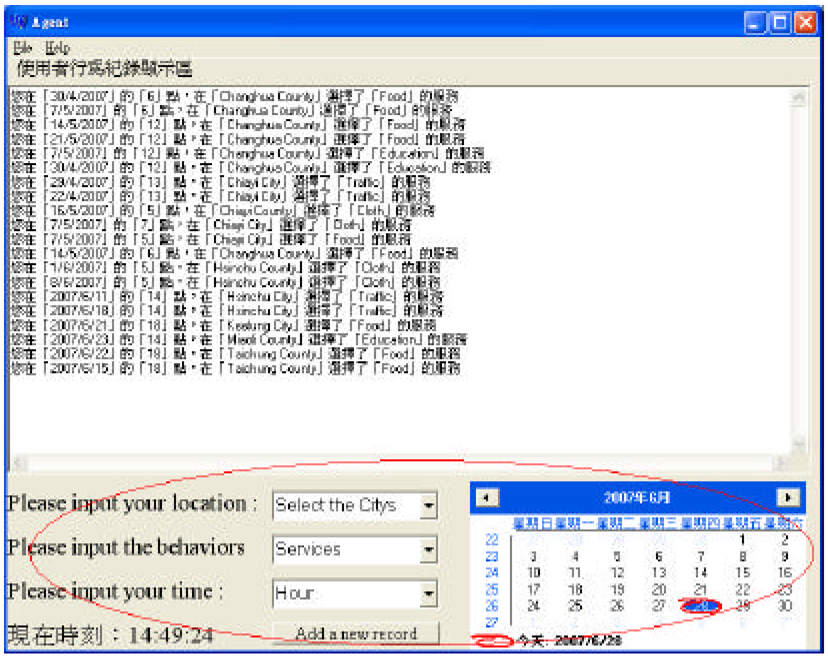
Figure 11: A template structure of a RSS2.0 seed

Take **Figure 7** for an example. After we have analyzed the user's shopping behaviors, we found that the user will need the food information in Taipei about 1:00 am in Taipei. The server will automatically search the RSS 2.0 seed generated from the store, and rebroadcast in personalized channel before 1:00. So, the agent combined RSS reader function will receive the seed, and notify the user of food information in Taipei before 1:00 am in Day1.

With this mechanism, users will not need to waste time searching for services in large databases, and we can let stores have more opportunities to publish their services.

Chapter 7 Tool Simulation

In this chapter, we build up a tool to simulate our research. At first, it is hard to collect the user log file from a really lifestyle website because we don't have a large lifestyle website. In order to solve it, we let users to input their shopping behavior, as the red circle of **Figure 12** shows.



```
您在「30/4/2007」的「6」點，在「Changhua County」選擇了「Food」的服務
您在「7/5/2007」的「6」點，在「Changhua County」選擇了「Food」的服務
您在「14/5/2007」的「12」點，在「Changhua County」選擇了「Food」的服務
您在「21/5/2007」的「12」點，在「Changhua County」選擇了「Food」的服務
您在「7/5/2007」的「12」點，在「Changhua County」選擇了「Education」的服務
您在「30/4/2007」的「12」點，在「Changhua County」選擇了「Education」的服務
您在「29/4/2007」的「13」點，在「Chiayi City」選擇了「Traffic」的服務
您在「22/4/2007」的「13」點，在「Chiayi City」選擇了「Traffic」的服務
您在「16/5/2007」的「5」點，在「Chiayi County」選擇了「Cloth」的服務
您在「7/5/2007」的「7」點，在「Chiayi City」選擇了「Cloth」的服務
您在「7/5/2007」的「5」點，在「Chiayi City」選擇了「Food」的服務
您在「14/5/2007」的「6」點，在「Changhua County」選擇了「Food」的服務
您在「1/6/2007」的「5」點，在「Hsinchu County」選擇了「Cloth」的服務
您在「8/6/2007」的「5」點，在「Hsinchu County」選擇了「Cloth」的服務
您在「2007/6/11」的「14」點，在「Hsinchu City」選擇了「Traffic」的服務
您在「2007/6/18」的「14」點，在「Hsinchu City」選擇了「Traffic」的服務
您在「2007/6/21」的「18」點，在「Keelung City」選擇了「Food」的服務
您在「2007/6/23」的「14」點，在「Miaoli County」選擇了「Education」的服務
您在「2007/6/22」的「18」點，在「Taichung County」選擇了「Food」的服務
您在「2007/6/15」的「18」點，在「Taichung County」選擇了「Food」的服務
```

Please input your location : Select the Citys

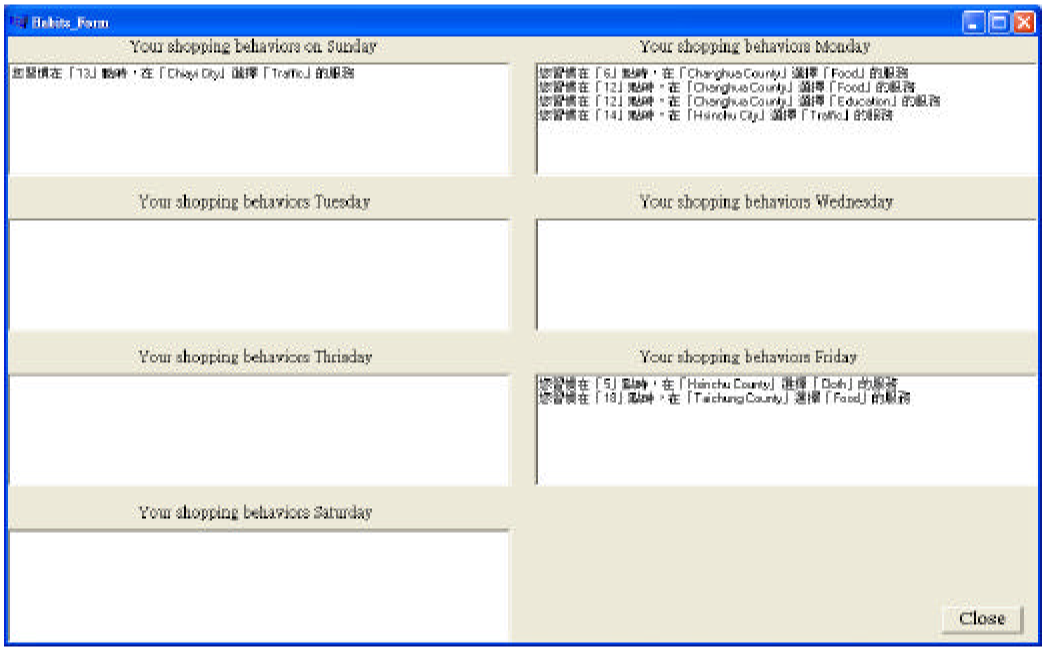
Please input the behaviors Services

Please input your time : Hour

現在時刻：14:49:24

Figure 12: The snapshot of the tool - 1

After the user input his behaviors, the agent will retrieve his shopping habits and show in **Figure 13**.



Your shopping behaviors on Sunday

您習慣在「13」點時，在「Chiayi City」選擇「Traffic」的服務

Your shopping behaviors Monday

您習慣在「6」點時，在「Changhua County」選擇「Food」的服務
您習慣在「12」點時，在「Changhua County」選擇「Food」的服務
您習慣在「12」點時，在「Changhua County」選擇「Education」的服務
您習慣在「14」點時，在「Hsinchu City」選擇「Traffic」的服務

Your shopping behaviors Tuesday

Your shopping behaviors Wednesday

Your shopping behaviors Thrisday

Your shopping behaviors Friday

您習慣在「5」點時，在「Hsinchu County」選擇「Cloth」的服務
您習慣在「18」點時，在「Taichung County」選擇「Food」的服務

Your shopping behaviors Saturday

Figure 13: The snapshot of the tool - 2

In order not to bother users work, the agent will hide in the right corner, as **Figure 14** shows.

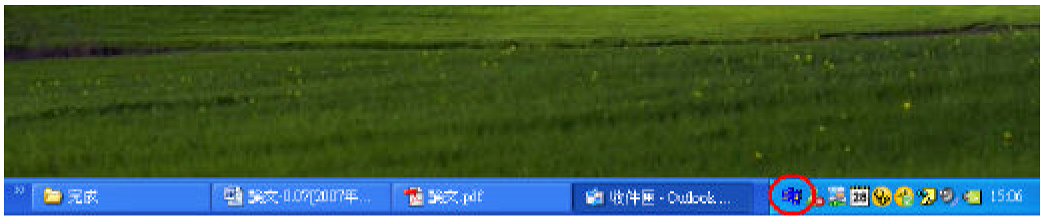


Figure 14: The snapshot of the tool - 3

If the user's habits will happen, the agent will automatically notify user, as **Figure 15** shows. When the user press this notification, the agent will pop a web browser program will the suitable services about user's behaviors, as **Figure 16** and **Figure 17** show.

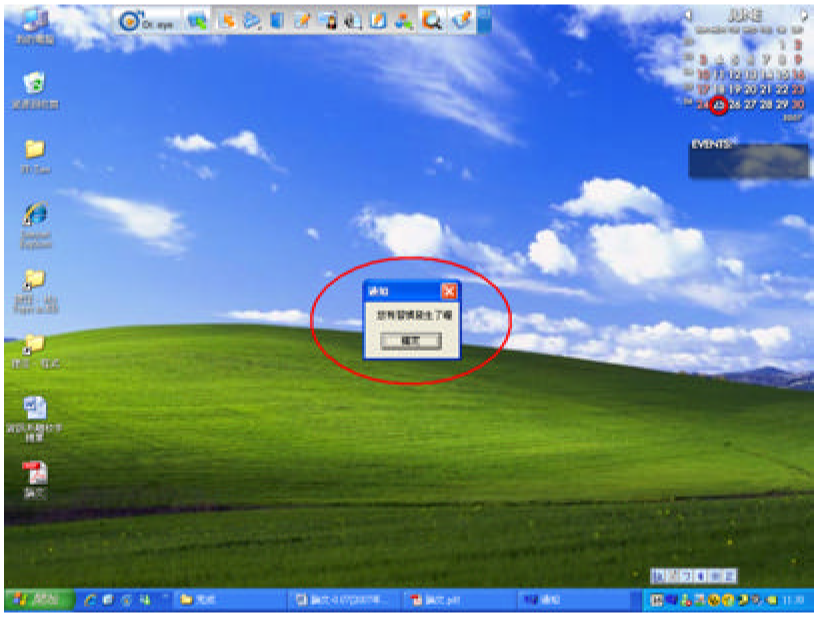
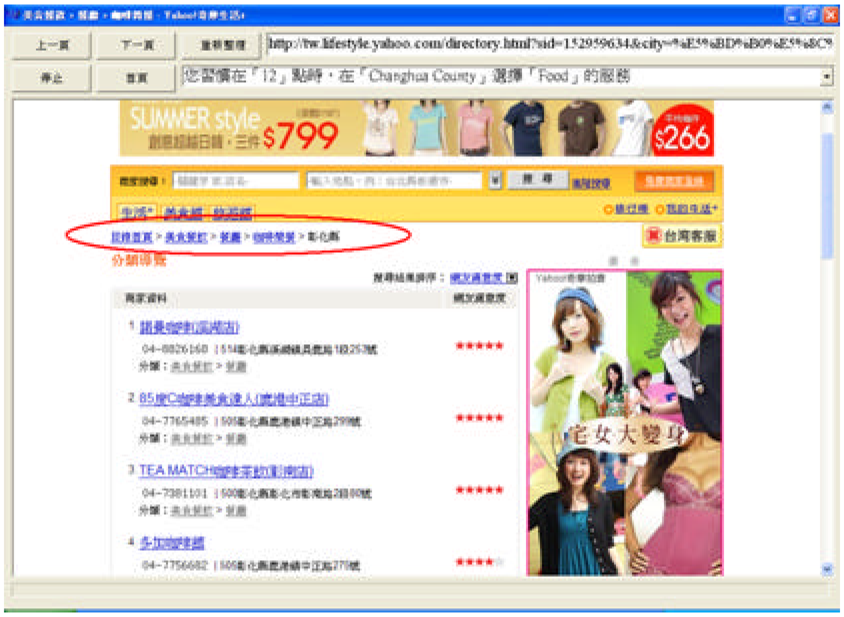


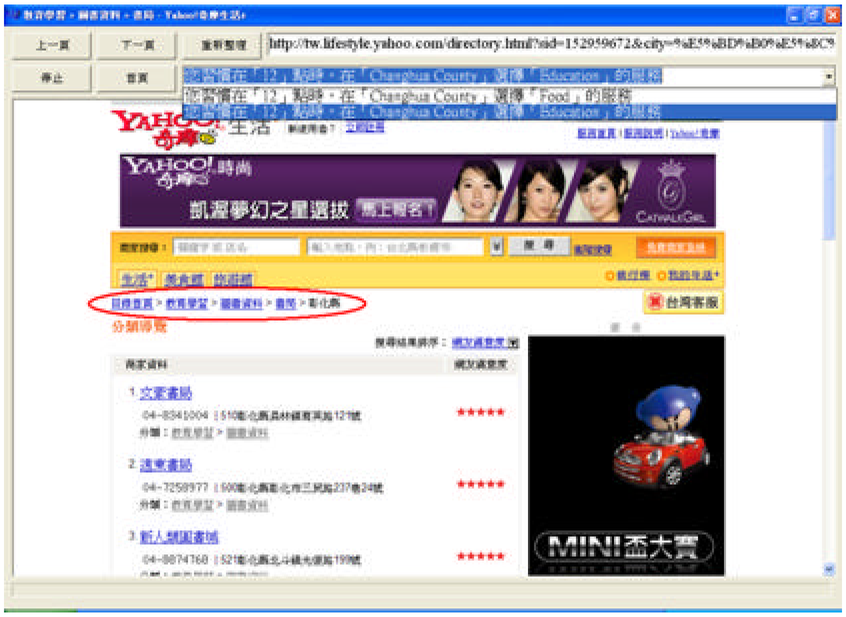
Figure 15: The snapshot of the tool - 4



http://tw.lifestyle.yahoo.com/directory.html?sid=152959634&city=%E5%BD%B0%E5%8C%

您習慣在「12」點時，在「Changhua County」選擇「Food」的服務

Figure 16: The snapshot of the tool - 5



http://tw.lifestyle.yahoo.com/directory.html?sid=152959672&city=%E5%BD%B0%E5%8C9

您習慣在「12」點時，在「Changhua County」選擇「Education」的服務

您習慣在「12」點時，在「Changhua County」選擇「Food」的服務

您習慣在「12」點時，在「Changhua County」選擇「Education」的服務

1 文豪書局

2 清華書局

3 新人類圖書城

Figure 17: The snapshot of the tool - 6

Note: behaviors in **Figure 16** and **Figure 17** are associated, so they will notify together at the same time.

In our tool simulation, the agent we proposed can not actively record user’s shopping behaviors because we do not have a large lifestyle website to collect user shopping behaviors. We only can record user’s shopping behaviors by user input them, and it’s a critical problem we need to solve in the future.

Chapter 8 Discussion

After integrating template cube and mining cube, we successfully retrieve the vertical habits. But, why we use the data cube-based structure? It is because that it has high extension. For example, we can change 24 segments to 48 or 12, and so do the location, category, and day. Moreover, not only the high extension, but it provides an efficient and effective structure for on-line analytical processing (OLAP) and on-line analytical mining.

Beside data cubed-based approach, we also use FP-Tree to find the horizontal behaviors. But we must remind that we use FP-Tree just because it is an efficient approach. If there is a new approach faster than it, we can adapt the new one to improve the prediction's efficiency. It is the most variable advantage of our high extensible structure.

Chapter 9 Conclusion and Future Work

We proposed an intelligent agent for the lifestyle website. The intelligent agent will not only analyze the user's shopping habits, but automatically notify users of suitable services before the habits happen.

In order to build the agent, we construct a personalized learning mechanism. The advantage of this mechanism is its high extension. In other words, we can easily modify and adapt to it. With this mechanism, users will not need to waste time searching for services in large databases, and we can let stores have more chances to publish their services.

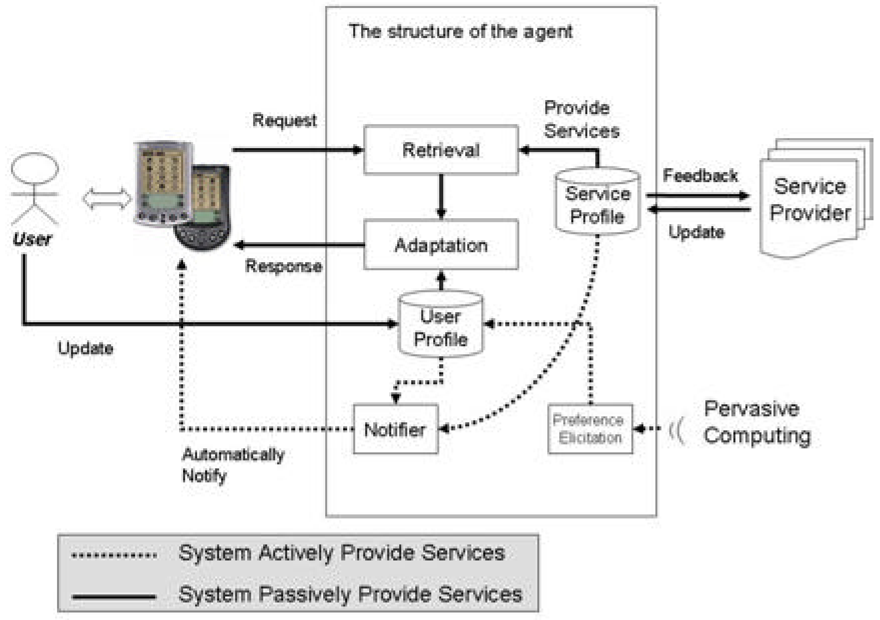


Figure 18: The structure of whole system in the future

As **Figure 18**, we hope to integrate pervasive computing to retrieve users' behaviors and construct a preference elicitation rule to elicit users' shopping habits. With this technique, we can automatically collect user habits and retrieve more shopping habits. With more accurate habits records, we can push more suitable services to users and let them feel more convenient.

Reference

[1] S. J. Soltysiak and I. B. Crabtree, "Automatic Learning of User Profiles — Towards the Personalisation of Agent Services," *BT Technology Journal*, Vol. 16, No. 3, pp.110-117, July 1998.

[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," *Proceedings of the 20th International Conference on Very Large Data Bases*, pp. 487–499, September 1994.

[3] R. Agrawal and R. Srikant, "Mining sequential patterns," *Proceedings of the 11th International Conference on Data Engineering*, pp. 3–14, March 1995.

[4] J Han, J Pei, and Y Yin, "Mining Frequent Patterns without Candidate Generation," *Proceedings of the 2000 ACM SIGMOD International Conference on Management of data*, pp. 1-12, 2000.

[5] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, 1998.

[6] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient Similarity Search in Sequence Databases," *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*, pp. 69-84, October 1993.

[7] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast Subsequence Matching in Time-Series Databases," *ACM SIGMOD Record*, Vol. 23, No. 2, pp. 419-429, June 1994.

[8] K. Chan and A. Fu, "Efficient Time-Series Matching by Wavelets," *Proceeding of the 15th International Conference on Data Engineering*, pp.126-133, March 1999.

[9] H. Mannila, H. Toivonen, and A. Verkamo, "Discovering Frequent Episodes in Sequences," *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pp. 210-215, 1995.

[10] D. Rafiei, "On Similarity-Based Queries for Time-Series Data," *Proceeding of 15th International Conference on Data Engineering*, 1999.

[11]J. Han, G. Dong, and Y. Yin, "Efficient mining of partial periodic patterns in time series database," *Proceeding of 15th International Conference on Data Engineering*, Sydney, Australia, Mar. 1999.

[12]Hammond, T, Hannay, and B. Lund, "The Role of RSS in Science Publishing," *D-Lib Magazine*, Vol. 10, No. 12, December 2004.

[13]J. Han, W. Gong, and Y. Yin, "Mining segment-wise periodic patterns in time-related databases," *Proceeding of 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'98)*, pp. 43-52, August 1998.

[14]http://cyber.law.harvard.edu/rss/rss.html

[15]S. Schiaffino and A. Amandi, "Polite Personal Agents," *IEEE Intelligent Systems*, vol. 21, no. 1, pp. 12-19, Jan/Feb, 2006

[16]R. J. Roiger and M. W. Geatz, Data Mining: A Tutorial-Based Primer, 2003.

[17]K. P. Sycara, "The Many Faces of Agents," *AI Magazine*, Vol. 19, no.2, pp. 11-12, 1998.

[18]M. Lewis, "Designing for Human – Agent Interaction," AI Magazine, Vol.19, No. 2, pp. 67-78, 1998.

[19]Y. Lashkari, M. Metral, and P. Maes, "Collaborative Interface Agents," *Proceeding of the Twelfth National Conference on Artificial Intelligence*, Menlo Park, CA: American Association of Artificial Intelligence, pp. 444-450.

[20]T. Payne and P. Edwards, "Interface Agents that Learn: An Investigation of Learning Issues in a Mail Agent Interface," *Applied Artificial Intelligence*, Vol. 11, No. 1, pp. 1-32, 1997.

[21]S. Haag, M. Cummings, and D. McCubbery, Management information systems for the Information Age, 3rd ed. Boston: McGraw-Hill, 2002.

[22]S. Case, N Azarmi, M. Thint, and T. Ohtani, "Enhancing E-Communities with Agent – Based System," *Computer*, pp. 64-69, July, 2001.

[23]E. H. Durfee, “Scaling Up Agent Coordination Strategies,” *Computer*, pp. 39-46, July, 2001.