Mining Survey Data

MAX BENNEDICH

KTH Computer Science and Communication

Master of Science Thesis
Stockholm, Sweden 2009
Mining Survey Data

MAX BENNEDICH

Master’s Thesis in Computer Science (30 ECTS credits) at the School of Engineering and Business Management Royal Institute of Technology year 2009
Supervisor at CSC was Henrik Eriksson
Examiner was Stefan Arnborg

TRITA-CSC-E 2009:007
ISRN-KTH/CSC/E---09/007--SE
ISSN-1653-5715

Royal Institute of Technology
School of Computer Science and Communication

KTH CSC
SE-100 44 Stockholm, Sweden

URL: www.csc.kth.se
Abstract

This thesis presents an efficient method for mining quantitative association rules for finitely discrete quantitative data. Our main contribution is a new view on what constitutes an unexpected relation, a view that is believed to be more accurate than previous models.

To ensure the statistical significance of the results, a non-parametric exact test is developed that does not require data to be normally distributed, works with sample sizes as low as a single sample and has a run time proportional to the sample size.

In addition to mining rules, we show how memory-based reasoning can be used to learn and to predict the users’ interests in rules. The method we suggest is compared to two other popular prediction methods, a Naive Bayes classifier and a neural network, and appears to be superior to both, both in terms of learning quickly and being able to reach a high prediction accuracy.

Data mining av enkätdata

Sammanfattning

Detta examensarbete presenterar en effektiv metod för att finna kvantitativa associationsregler ur ändligt diskret kvantitativ data. Vårt främsta bidrag är en ny definition av vad som utgör en oväntad relation, en definition vi tror är mer precis än tidigare modeller.

För att säkerställa den statistiska signifikansen av resultaten har ett icke-parametriskt exakt test utvecklats, som inte kräver att data är normalfördelat, fungerar med stickprovsstorlekar så låga som ett enda element och har en kortid proportionell mot stickprovsstorleken.

I tillägg till att söka efter regler, visar vi hur minnesbaserat resone-mang kan användas för att lära sig att förutsäga användares intresse för regler. Metoden vi föreslår jämförs med två andra populära klassificeringsmetoder, en Naive Bayes-klassifikator och ett neuralt nätverk, och verkar prestera bättre än bägge, både i termer av snabbare inlärningsförmåga och högre säkerhet i uppskattningen.
# Contents

1 **Introduction**  
  1.1 Purpose ................................................. 1  
  1.2 Limitations and assumptions ............................ 2  
  1.3 Disposition ........................................... 2  

2 **Preliminaries**  
  2.1 Statistics ............................................. 3  
    2.1.1 Hypothesis testing ................................. 3  
    2.1.2 Multiple testing .................................. 4  
    2.1.3 Resampling ........................................ 4  
  2.2 Surveys ............................................... 5  
    2.2.1 Types of survey questions .......................... 5  
    2.2.2 Analyzing survey data ............................... 6  
    2.2.3 Missing data .................................... 8  
  2.3 Data mining .......................................... 9  
    2.3.1 The idea of data mining ............................ 9  
    2.3.2 Association rules ................................. 9  
    2.3.3 Neural networks .................................. 10  
    2.3.4 Decision trees ................................... 10  
    2.3.5 Clustering ....................................... 11  
    2.3.6 Memory-based reasoning ............................ 11  

3 **Related work**  
  3.1 Clustering ........................................... 13  
  3.2 Association rules ..................................... 14  
    3.2.1 The Apriori algorithm .............................. 14  
    3.2.2 Mapping quantitative to binary attributes ....... 14  
    3.2.3 Impact rules ...................................... 15  
    3.2.4 Large and dense data sets ......................... 16  
  3.3 OLAP cubes and exceptions ............................. 17  
  3.4 Rule caching ......................................... 19  
  3.5 Interestingness ....................................... 19
### 4 Mining survey data

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Method selection</td>
<td>23</td>
</tr>
<tr>
<td>4.2</td>
<td>Data</td>
<td>24</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Available data</td>
<td>24</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Preparation of data</td>
<td>25</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Missing data</td>
<td>26</td>
</tr>
<tr>
<td>4.3</td>
<td>Quantitative association rules</td>
<td>27</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Finding itemsets</td>
<td>27</td>
</tr>
<tr>
<td>4.3.2</td>
<td>An unexpectedness model</td>
<td>28</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Statistical test</td>
<td>29</td>
</tr>
<tr>
<td>4.4</td>
<td>Trends</td>
<td>34</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Trends as quantitative association rules</td>
<td>34</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Trend interestingness</td>
<td>35</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Finding trends efficiently</td>
<td>35</td>
</tr>
<tr>
<td>4.4.4</td>
<td>Testing trends statistically</td>
<td>36</td>
</tr>
<tr>
<td>4.5</td>
<td>Rule cache</td>
<td>37</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Trends and the rule cache</td>
<td>37</td>
</tr>
<tr>
<td>4.6</td>
<td>Presentation</td>
<td>37</td>
</tr>
<tr>
<td>4.7</td>
<td>Summary</td>
<td>39</td>
</tr>
</tbody>
</table>

### 5 Interestingness

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Subjectivity</td>
<td>41</td>
</tr>
<tr>
<td>5.2</td>
<td>A model</td>
<td>41</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Rule components</td>
<td>41</td>
</tr>
<tr>
<td>5.2.2</td>
<td>The method by Xin et al.</td>
<td>42</td>
</tr>
<tr>
<td>5.2.3</td>
<td>Other models</td>
<td>43</td>
</tr>
<tr>
<td>5.3</td>
<td>Memory-based reasoning</td>
<td>43</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Prediction of interestingness</td>
<td>43</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Certainty of prediction</td>
<td>44</td>
</tr>
<tr>
<td>5.4</td>
<td>Rule distance</td>
<td>45</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Existing methods</td>
<td>45</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Distance and similarity</td>
<td>45</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Similarity between quantitative attributes</td>
<td>46</td>
</tr>
<tr>
<td>5.4.4</td>
<td>Similarity between categorical attributes</td>
<td>46</td>
</tr>
<tr>
<td>5.4.5</td>
<td>Multiple categorical attributes</td>
<td>47</td>
</tr>
<tr>
<td>5.4.6</td>
<td>Refining the similarity measure</td>
<td>49</td>
</tr>
<tr>
<td>5.5</td>
<td>User input</td>
<td>49</td>
</tr>
<tr>
<td>5.5.1</td>
<td>Rating versus ranking rules</td>
<td>49</td>
</tr>
<tr>
<td>5.5.2</td>
<td>Clustering and optimal learning</td>
<td>50</td>
</tr>
<tr>
<td>5.6</td>
<td>Summary</td>
<td>50</td>
</tr>
</tbody>
</table>

### 6 Results

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Finding exceptional phenomena</td>
<td>53</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Data</td>
<td>53</td>
</tr>
</tbody>
</table>
6.1.2 The experiment .............................. 53
6.1.3 Results ..................................... 54
6.2 Scalability ...................................... 56
   6.2.1 The experiment ............................ 56
   6.2.2 Results for authentic data ................... 57
   6.2.3 Results for synthetic data .................... 57
6.3 Learning the user’s interests ..................... 61
   6.3.1 A user model .............................. 61
   6.3.2 The experiment ............................ 61
   6.3.3 Results .................................. 62
6.4 Feedback model .................................. 63
   6.4.1 The experiment ............................ 63
   6.4.2 Results .................................. 64

7 Conclusions ................................. 67
   7.1 Generalization of work ....................... 67
   7.2 Fulfillment of purpose ....................... 68
   7.3 Future work ................................. 68

Bibliography ................................. 71
Key concepts in customer relationship management are those of knowing the customer and satisfying the customer. An efficient method to get to know the customer is to use surveys to collect information. Once the information has been collected, it can be analyzed in various ways to reveal insights about the customers and their needs. Typical tools include dividing respondents into categories and using graphs, tables and statistical measures to summarize and better understand vast amounts of data. For example, customer satisfaction can be plotted over time to reveal trends and mean satisfaction levels can be calculated and compared in between categories.

Today it is not uncommon to collect large amounts of customer information through surveys, perhaps millions of records. In addition to the type of information that can be found with the methods mentioned above, such a database is certain to contain more elusive patterns and relations that can be of business use. The problem of analyzing large amounts of data to find hidden and previously unknown information is commonly referred to as data mining.

1.1 Purpose

The purpose of this thesis is to find an effective solution for discovering non-trivial trends and relations out of a large amount of satisfaction surveys, i.e. a solution for mining survey data. The solution is primarily intended to be used by large chain companies, for example with guests at hotel chains, gym chains and car rental chains, and the goal is to find relations that can be of business use. One example of a relation is if a particular group of customers at a certain facility (e.g. a certain hotel) is unexpectedly satisfied/dissatisfied with some specific area. Another example is occurring or time related trends, such as if the customer satisfaction at a certain hotel has dropped the last two weeks or drops on certain weekdays.

It is desirable that the developed solution is as general and automatic as possible, so that it need not be redeveloped for each business and so that the results can be presented to decision makers directly without the need for an analyst to interpret or sort them first. Great effort must therefore be put into determining which relations
are interesting to the user and which are uninteresting, obvious or even spurious. It is important that the solution can be integrated into an existing framework and work continuously with the data so that the relations that are presented are up-to-date (a daily update rate of the relations is desired).

1.2 Limitations and assumptions

Throughout the thesis, it will be assumed that the surveys are stored in digital form and contain demographic, behavioral, attitudinal (respondent satisfaction for various factors on a numeric scale, e.g. 1-10) and free text data. It is further assumed that the set of customers for whom survey data is available is a random sample of all the customers. This assumption is likely to be flawed, since the opinions of customers who choose to respond to the survey may differ from those of non-respondents. No effort will be made here to analyze the response rate or to suggest how it can be increased. Furthermore, the design of the survey and the entire survey process is considered outside the scope of this thesis. Thus, the starting point of this project is the available survey data and the assumption that every effort has been made to make it as valid and reliable as possible.

Computer analysis of free text data, so called text mining, is a very rewarding field of research, but will not be considered in this thesis.

1.3 Disposition

This thesis is organized as follows. First, a few preliminary, general topics are covered in chapter 2. In chapter 3 we describe how problems similar to ours have been dealt with by other researchers. Chapters 4 and 5 describe our contributions to the field, the former chapter concerns the data mining process and the latter how to determine which results are interesting to a user. In chapter 6, experiments are performed with the suggested method and it is evaluated and discussed regarding performance and scalability. Finally, chapter 7 concludes the project and suggests future work.

Several examples will be given in this thesis in order to simplify the understanding of methods and problems. Even though the methods described are intended to work for a wide range of businesses, the examples given exclusively concern the hotel business for reasons of consistency.
Chapter 2

Preliminaries

This chapter is intended to give the reader a brief background of a few general topics that form important parts of this project: statistics, surveys and data mining. The aim is not to provide a complete overview of these topics – for this, other sources are referred to – but it is hoped that the information given is enough for the thesis to be comprehended on its own. This chapter will also establish some of the terminology used throughout the thesis.

2.1 Statistics

This section will give a short overview of a few statistical concepts used in this thesis. For a more thorough introduction to statistics, see for example [26].

2.1.1 Hypothesis testing

Given a set of observations, and a hypothesis about these, we may use a test to assess the validity of the hypothesis. Hypotheses often concern parameters of a population, for example mean values. The test is set up by first formulating a null hypothesis $H_0$, seen as the default hypothesis, e.g. stating that the mean values of two populations are equal and that any observed difference is due to happenstance. Similarly, an alternative hypothesis $H_1$ is set up, e.g. stating that population means are not equal.

A test statistic is then chosen depending on the hypothesis, and an associated $p$-value $p$ is calculated with the help of a probability model. A $p$-value is the probability that, assuming that the null hypothesis is true, the test statistic would be at least as extreme as the one obtained just by chance. The lower $p$ is, the more reason one has to doubt the validity of the null hypothesis.

A significance level $\alpha$ is commonly used to determine whether to reject $H_0$ or not, often $\alpha$ is set to 0.05 (5%). If $p \leq \alpha$, $H_0$ is rejected, suggesting that $H_1$ may be true. Such a result is said to be statistically significant. If $p > \alpha$, $H_0$ is not
rejected. This does not mean that $H_0$ is true, but simply that there is not enough evidence to reject $H_0$.

Two types of errors are used together with hypothesis testing: type I and type II error. A type I error is to reject $H_0$ when it is in fact true while a type II error is to not reject $H_0$ when it is in fact false. A type I error is usually considered more serious than a type II error, as accepting $H_1$ may prompt wrong actions. It can easily be seen that the probability of a type I error is $\alpha$. The probability of a type II error depends on the power of the test that is used and is generally unknown. For many tests it can be reduced by increasing the sample size.

### 2.1.2 Multiple testing

Hypothesis testing is frequently used to test a certain theory or hypothesis that a researcher has, in which case $\alpha = 0.05$ may be an acceptable level of significance. Nonetheless, when several hypothesis tests are carried out, it is expected for some tests to obtain low $p$-values just by chance, and the probability of committing a type I error in at least one of the tests – referred to as the experimentwise error rate – grows larger than $\alpha$. For example, if we are testing a hypothesis with 10 independent populations, and $\alpha = 0.05$ is used for each test, the experimentwise error rate is $1 - (1 - 0.05)^{10} \approx 40\%$.

When performing multiple tests, it is therefore usually desired to modify the significance level used in each test so as to keep the experimentwise error rate equal to $\alpha$. The simplest, and also the most conservative, method of doing this is the Bonferroni correction, which uses the level $\alpha/n$ where $n$ is the number of tests performed. If the hypotheses tested are not independent of each other, the situation becomes much more complicated. For more information about multiple testing, consult for example [24].

### 2.1.3 Resampling

The probability models used in hypothesis testing often make the assumptions that the data is being distributed normally or that the sample sizes are large. When this is not the case, another method, called resampling, can be used to more accurately compute the $p$-value of a test statistic. The idea of resampling is to repeatedly draw samples from the population being studied, generating a distribution based on the actual data rather than on a parametric representation. The drawback of resampling is that, although the mathematics involved is elementary, it uses more computer resources than a parametric test since thousands of resamples may be required. However, with computers constantly getting both faster and cheaper, resampling is gaining popularity [26].

Resampling can be performed in a few different ways. One way, known as bootstrapping, is to draw samples with replacement from the population. The rationale is that the population being studied can be seen as a representation of a general (infinite) population, so resampling from the studied population corresponds to
2.2. SURVEYS

sampling from the general population. Another method is known as permutation testing, during which samples are drawn without replacement. This test corresponds to labeling data as being part of either the sample or the population and then studying the test statistic as the labels are permuted. An important observation is that this requires that the distributions of the sample and the population are identical when the null hypothesis is true.

Resampling can either be done exactly, by enumerating and analyzing every possible way to draw samples from the population, or approximated with a Monte Carlo method by testing a random selection of all possible rearrangements. Whether to use the bootstrap or the permutation approach to compute \( p \)-values depends on the problem. The permutation test is more commonly seen in the literature, but in [32], Simon argues in favor of bootstrapping. He states that sampling with replacement is more appropriate if the size of the general population is not assumed to be fixed and if data are assumed to be independent of each other. When sampling without replacement, as samples are drawn, the distribution of the remaining samples change, implying a finite sized and specific population.

2.2 Surveys

2.2.1 Types of survey questions

Mitchell and Jolley [25] categorize survey questions into three basic formats: nominal-dichotomous, Likert type and open-ended. The first type of question is used to group respondents into categories, for example according to gender, ethnicity or purpose of visit, as exemplified in the left hand side of figure 2.1. Dichotomous questions refer to questions with only two alternatives and are treated the same way as multiple-choice questions. This type of question yields nominal data, i.e. data representing different states among the respondents. Mitchell and Jolley state that nominal data is considered the least informative type of data since it lacks a way to be ordered meaningfully. In this thesis, we will refer to nominal-dichotomous variables as categorical attributes, which assume one of several possible categories.

![What is your purpose of visit?](image)

- Business
- Leisure
- Conference
- Sports event
- Family visit
- Other

I was satisfied with the pool
I was satisfied with the breakfast

Disagree ==> Agree

Figure 2.1. An example of a nominal survey question (left) and a Likert type question (right).
Likert type questions allow the respondents to state their opinions using a scale, for example a 5 point scale where 1 equals “Strongly disagree”, 2 means “Disagree”, 3 is “Neutral”, 4 is “Agree” and 5 is “Strongly agree”, as shown in the right hand side of figure 2.1. To be correct, Likert questions result in ordinal data; it is possible to say that a respondent answering 5 agrees more than one answering 4, but it cannot be said that the difference between 5 and 4 is as large as the difference between 3 and 2, although both are separated by 1 point on the scale. However, according to Mitchell and Jolley, most psychologists work under the assumption that equal distances on a Likert scale correspond to equal psychological distance. That way, the data can be used as interval data and a much wider range of statistical tools are available to analyze it. We will be referring to interval variables as quantitative attributes.

Open-ended questions let the respondents reply in their own words. The advantages of this type of question are that it can give very specific and detailed responses, and it avoids the bias of giving the respondent a set of fixed responses. The main disadvantage is that open-ended questions are difficult to analyze, especially when done by a computer.

2.2.2 Analyzing survey data

A simple way to analyze survey data is to summarize it in different ways. For a mixture of categorical and quantitative data, the mean values of the quantitative attributes can be calculated and summarized for the different categorical attributes. Inferential statistics can be used to test whether an observed difference between categories represents an actual difference in the general population, c.f. section 2.1.1.

If a researcher wishes to study the relationship between two or more categories and a quantitative attribute at once, the situation becomes more complicated. Although this type of analysis is not an experiment in the scientific sense, the same methodology as for experiments can be used in the case that the categorical data is of independent nature (e.g. demographical and behavioral data) and the quantitative data is of dependent nature (i.e., affected by the independent variables). Since the categorical data is not assigned randomly, they are called weak independent variables, and the study is referred to as a nonexperiment. It is not possible to conclude that a weak independent variable is the cause of an effect observed in a dependent variable. For example, given the variable gender and the observation that females are on average less satisfied with a hotel stay, it is not possible to say that gender alone has anything to do with customer satisfaction. There may be an abundance of ways that female guests on average differ from male ones – age, occupation, expectations, previous hotel experiences, psychological differences, etc. – and it cannot be determined whether any of these factors is a more legitimate reason than gender. Although this is a dilemma for researchers, it need not be for businesses. It may be sufficient for a business to determine that females are less satisfied, and use that knowledge to aim at improving their satisfaction regardless
2.2. SURVEYS

of what the underlying reason is.

Each independent variable is said to have a main effect on the dependent variables, and similarly, each combination of independent variables may have an interaction effect. Main and interaction effects are very important topics for this project and will be explored by means of a few examples. All examples concern a hotel environment and a 10 point Likert scale survey question asking the guests to rate the food quality of the hotel. Here, 1 means that a guest is very dissatisfied and 10 means very satisfied. Each group in the examples shows the mean value for all the respondents pertaining to that group. The number of respondents in each group is ignored at this point, but is a topic that will be revisited in section 3.3.

Let us first consider the categorical attributes \textit{Hotel} and \textit{Breakfast}, signifying what hotel the guest stayed at and whether the guest ate breakfast at the hotel. For the hotel attribute, we will use the categories A for the guests that stayed at hotel A and non-A for all other guests. For the breakfast attribute, we use the categories Yes and No. This arrangement gives rise to a 2x2 design as follows:

<table>
<thead>
<tr>
<th>Hotel = non-A</th>
<th>Hotel = A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breakfast = No</td>
<td>6.8</td>
</tr>
<tr>
<td>Breakfast = Yes</td>
<td>8.1</td>
</tr>
</tbody>
</table>

The overall main effect of staying at hotel A is defined as the average of its two simple main effects, i.e. the differences of the groups who stayed at hotel A and the groups that did not. Thus, the main effect of staying at hotel A is the average of (7.5−6.8) and (8.8−8.1), which equals (0.7+0.7)/2 = +0.7 units. Similarly, the main effect of having breakfast is ((8.1−6.8)+(8.8−7.5))/2 = (1.3+1.3)/2 = +1.3 units. In this case, the attributes are completely independent of each other and display no interaction. Regardless of whether a guest had breakfast or not, the effect of staying at hotel A is still +0.7 units, and regardless of whether a guest stayed at hotel A or not, the effect of having breakfast is +1.3 units. The sum of the main effects is 0.7 + 1.3 = 2.0 units, which is the difference in mean between the groups <\textit{Hotel} = non-A, \textit{Breakfast} = No> and <\textit{Hotel} = A, \textit{Breakfast} = Yes>. An explanation for this example could be that the standard at hotel A is higher than average, making guests on average 0.7 units more satisfied in all areas, and that breakfast tastes just as good at hotel A as any other hotel.

As another example, let us consider the attributes \textit{Dinner} and \textit{Breakfast} for some set of guests:

<table>
<thead>
<tr>
<th>Dinner = No</th>
<th>Dinner = Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breakfast = No</td>
<td>7.5</td>
</tr>
<tr>
<td>Breakfast = Yes</td>
<td>8.6</td>
</tr>
</tbody>
</table>

The main effect of having dinner is (1.2 + 0.4)/2 = +0.8 units and the main effect of having breakfast is (1.1 + 0.3)/2 = +0.7 units. This time, however, there is a
clear interaction between the attributes. Guests that had breakfast are not much affected by also having had dinner and vice versa. A possible explanation for the interaction is that the two attributes express similar things: guests who eat at the hotel are more satisfied with its food. So for a guest that already had dinner, the breakfast does not make much of a difference. The attributes are in this case not independent of each other.

An alternative explanation is what Mitchell and Jolley refer to as the ceiling effect, which may occur when the measure is not capable of recording responses above a certain level. With such a high mean value for the group having dinner but no breakfast, it is likely that many of those guests are responding with the maximum value (i.e., 10). If such a guest would have had breakfast too and had become even more satisfied with the food quality, it would not have been possible for him to respond any higher. The same effect exists in the other direction with responses close to the minimum value and is called the floor effect. With either of these effects, the interaction is spurious and a result of the measure limiting the respondents from answering what they would have wanted.

As a final example, consider the same attributes and the following design:

<table>
<thead>
<tr>
<th></th>
<th>Dinner = No</th>
<th>Dinner = Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breakfast = No</td>
<td>6.2</td>
<td>3.4</td>
</tr>
<tr>
<td>Breakfast = Yes</td>
<td>7.6</td>
<td>3.6</td>
</tr>
</tbody>
</table>

The main effect of having dinner is \((-2.8 + -4.0)/2 = -3.4\) units and the main effect of having breakfast is \((1.4 + 0.2)/2 = +0.8\) units. This time, the interaction is even larger than in the previous example. An explanation could be that the breakfast is good while the dinner is more towards awful, and guests that have both are more prone to recall the bad dinner. Again, the attributes are not independent of each other, this time due to one attribute being stronger than and dominating the other.

### 2.2.3 Missing data

There are several ways to handle missing data fields in a survey [6]. One way, which may be appropriate if only a few of the surveys are incomplete, is to simply exclude those surveys from the analysis. Alternatively, missing values can be omitted on an individual basis. These methods typically introduce bias since missing data is unlikely to be distributed randomly.

Another commonly used method is to replace a missing value with the mean or the most common value of that attribute. It is easily seen that such an approach can lead to spurious results; imagine that a survey asks about a respondent’s age, gender and height, and one male respondent aged 7 has omitted the height. Using the mean value of, say, 167 cm is clearly not a good solution. More elaborate methods infer missing values from the circumstantial data. For the example above, there may be 10 other male respondents aged 7, and using the height of one of them picked at random may yield the value 118 cm.
2.3 Data mining

2.3.1 The idea of data mining

Data mining can be defined as the search for non-trivial and useful patterns and relations out of a large amount of data. From a commercial standpoint, the goal of data mining is usually to increase the knowledge about a corporation’s customers so that the corporation can increase its sales. There are numerous different data mining techniques, and which one to use depends on what one is looking for, in which form one wants the results and the format and amount of available data. Data mining tasks can be divided into two groups: directed or undirected.

In a directed task, an analyst has a clear idea of what he is looking for: to explain or predict one or more specific output variables based on a set of input variables. For example, he may wish to classify a set of customers into potential buyers and non-buyers based on their demographics. To solve such a task, a supervised learning approach is used where the data mining application is given a training set for which both the input and output is known.

In an undirected task, all variables are treated as input variables and no training set is used, hence the learning approach is said to be unsupervised. An analyst may use undirected data mining to find customers that are similar in some aspect, or customer groups that show an unexpected behavior, without really knowing what he is looking for.

For a more extensive overview of data mining and knowledge discovery, see for example [10]. A more technical overview, focused on association rules, is found in [36]. A good source of how data mining can be applied in a business environment is [6].

2.3.2 Association rules

Association rules, or market basket analysis as it is sometimes called, is an unsupervised method that was invented to search for and describe relations between items in a database of sales transactions. The technique is used to answer questions such as which items sell best, which items usually sell together and how the customer’s demographics affect their purchasing behavior. For example, an association rule could state that when a customer buys bread and milk, he is likely to also buy eggs. A rule is usually accompanied by its support (i.e. the percentage of all the transactions that contain the items bread, milk and eggs) and its confidence (i.e. the percentage of the transactions containing the items bread and milk that also contain the item eggs), and possibly by other statistical measures to denote how unexpected the rule is. Usually, a rule with too low support is considered uninteresting since it is too rare to be of business use while a rule with too low confidence is considered uninteresting since it is not a very strong rule. Since the introduction of association rules, they have been applied in many other fields, for example in the search for words that appear together unusually frequently in a document [17], and in the analysis
Chapter 2. Preliminaries

of sequential data to locate events that frequently occur shortly after another set of events [23].

Advantages of association rules are that the rules are easy to understand and that one is not required to specify beforehand what to look for. A disadvantage is that a large amount of rules may be generated, of which only a few are truly useful, and it may be difficult to algorithmically determine which rules are interesting to the user.

2.3.3 Neural networks

A neural network is based on the biological model of how a brain works, and is made up of artificial neurons that are connected together. The point of most neural networks is to map a number of input variables to one or more output variables. They can be used as both a directed and undirected method. Most common is to use supervised learning, where a number of examples of inputs and outputs are presented to the network, and it tries to adapt itself to map the inputs to the outputs as well as possible. Tasks that can be solved with supervised learning include classification and prediction problems, pattern recognition and function approximation. With unsupervised learning, the network uses only the input data to adapt itself. Example applications of this method are clustering and dimensionality reduction.

A large drawback of neural networks is that the results are difficult to motivate. Many other data mining techniques can provide a list of rules or a statistical justification for their results; neural networks on the other hand justify their results by obscure layers of weights and complicated mathematical formulas. Another problem, as pointed out in [6], is that it may take some effort to get a neural network to work optimally, since there are many parameters that may be adjusted. There is also a risk that a network becomes overtrained and memorizes the example data rather than generalizing from it.

2.3.4 Decision trees

Decision trees pertain to the class of supervised learning methods, and is an alternative to neural networks when it comes to classification tasks or prediction of discrete values. A decision tree consists of a sequence of rules that divides the population into smaller and smaller groups. For each division, the groups become more homogeneous with respect to the class or variable that is to be predicted. Figure 2.2 shows an example of using a decision tree to predict whether a guest staying at a hotel will return for another stay within the next six months. The strength of decision trees is that, contrary to neural networks, they are easy to read and understand, and the predictions can be readily motivated. A disadvantage is that they are not suited at predicting continuous values. Compared to association rules, whose rules are akin to those mentioned here, decision trees require that one specifies beforehand which variable to predict.
2.3. DATA MINING

![Decision Tree Diagram]

Figure 2.2. An example of a decision tree used to predict whether a hotel guest will return within the next six months.

2.3.5 Clustering

Clustering is a popular tool in order to gain an understanding of and describe a set of data. It uses unsupervised learning; given a set of data, it partitions the data into clusters (groups) so that the members of each cluster are similar in some way. Clustering can be used with any kind of data as long as the distance (or, equivalently, the similarity measure) between two instances is defined. Distance metrics is a well studied problem, and distance functions exist for a wide variety of instances; points in Euclidean space, web pages and association rules to mention just a few.

Several different clustering methods exist depending on the application. For example, the number of clusters can be decided beforehand, or determined automatically; clusters can form hierarchies, or separate partitions; the results can be determinate, or random at each run. As a data mining tool, clustering is commonly used to segment a population into representative customers groups.

2.3.6 Memory-based reasoning

Memory-based reasoning, or MBR, is a nearest neighbor technique employing supervised learning and used for classification or prediction problems. Its name refers to how humans tackle new problems by using experiences and memories from similar past events. MBR uses the same idea: Given an unsolved problem instance, similar problems (neighbors) that have already been solved are located, and their information is combined to come up with a suggestion to the new problem. MBR does not create a model of the data as many other data mining techniques do - instead, the model can be thought of as being the data itself, and the system becomes more and more accurate as more problems are solved.

The two main concepts of MBR are how to measure the distance between instances, so that the nearest neighbors can be located, and how their information should be combined. Regarding distance metrics, the same methods can be used as for clustering, c.f. section 2.3.5. In [6], Berry and Linoff mention a few different
ways to combine information from neighbors. For prediction tasks, they suggest using a weighted average of the value of the $k$ nearest neighbors, where the weight is inversely proportional to the distance. Results are easy to motivate and MBR is equally well suited to discrete and continuous outputs.
Chapter 3

Related work

This chapter describes how past researchers have chosen to tackle problems similar to the one dealt with in this thesis. In chapter 4 and 5, many of their ideas will be further commented upon as they are interpreted in the context of our problem.

3.1 Clustering

In [21], Jing encounters a problem that resembles ours as he wishes to analyze a set of students described by a mixture of categorical and quantitative attributes. The categorical data is mainly of demographic nature while the quantitative data consists of various measures of the students’ achievements: grades, number of courses taken, etc. Jing describes how clustering of the quantitative data can be used in order to identify six types (clusters) of students: one type that studies hard and performs well, another type that fails most courses, etc. When these clusters have been found, the demographic attributes are used in order to analyze what each cluster consists of. This is mostly done graphically with the help of distribution bar graphs.

Jing observes that the finding of these clusters is not a push-button process, but an interactive process requiring careful attention by an analyst. Jing starts by analyzing which of the quantitative attributes to use for clustering, and decides for three key attributes. He then selects a clustering method and experiments to find how many clusters should be used to give a meaningful result. In this iterative process, results are evaluated by computing means over the clusters and by the use of graphical tools such as animated three-dimensional plots (for his three chosen attributes).

In [1], Agrawal et al. argue that conventional clustering algorithms fail for mining high dimensional data. They state that as the number of dimensions increase, distance functions become less useful since the density of points is likely to become low throughout the data space. This in turn prevents the detection of clusters existing in subspaces of the high dimensional data. The authors proceed by presenting an algorithm that is able to find such clusters. Briefly put, the first step is to discretize each dimension into a number of intervals, e.g. 10 intervals. They then start
the search by looking for clusters existing in any single dimension. Next they look for clusters existing in any combination of two clusters, then any combination of three clusters, etc., using a procedure that is very similar to the Apriori algorithm that will be described in section 3.2.1. Consult [1] for the full algorithm.

3.2 Association rules

3.2.1 The Apriori algorithm

Agrawal, Imielinski and Swami first introduced the concept of association rules in [2] in 1993. The following year, Agrawal and Srikant published a method called Apriori in order to solve the association rule problem more efficiently [3]. Both of these papers concern finding association rules in a large database of sales transactions. For each transaction, each item has a binary state: either present (1) or absent (0). A transaction can then be seen as a list of attributes taking the value 1 or 0.

The first step of the Apriori algorithm is to locate all sets of items having at least a certain minimum support; these sets are referred to as large itemsets. This is done by first locating all large individual items. These are merged to form a list of candidates for large pairs of items, and the database is then scanned again to check whether each of the candidates is indeed large. The large pairs are then merged to form a list of candidates for large triplets, the database is scanned again, etc. This cycle continues until itemsets of all sizes have been found. In a second step, the list of generated itemsets and a user specified minimum confidence level are used to create a list of association rules. For a more detailed description please see [3].

Several methods for finding large itemsets more quickly than Apriori have been invented. Many are based on Apriori but aimed at reducing the number of scans of the database. Another popular method is the FP-tree (frequent pattern tree) algorithm by Han et al. [14] that avoids the generation of candidates and is about an order of magnitude faster than Apriori.

3.2.2 Mapping quantitative to binary attributes

In [33], Srikant and Agrawal introduce the problem of mining quantitative data. They consider finding quantitative association rules, i.e. associations in relational tables with quantitative attributes. They wish to find relations of the type

10% of married people between the ages 50 and 60 have at least 2 cars

or, written another way

Married = Yes, Age ∈ [50,59] ⇒ Cars ≥ 2 for 10% of the population

In this case the individual attributes of each instance can be either quantitative (e.g. age or income) or categorical (e.g. marital status or zip code). Srikant and Agrawal describe how this problem can be solved by first mapping all attributes to binary attributes, for example by mapping the age attribute to a series of binary
3.2. ASSOCIATION RULES

attributes $<Age \in [20,29]>$, $<Age \in [30,39]>$, etc., of which one takes the value 1 and the others 0. After this stage, the standard Apriori algorithm can be used with the mapped data.

The authors describe how this simple mapping method can lead to a conflict of objectives, where on the one hand one needs to create intervals large enough to have sufficient data; if the intervals were $<Age \in [20,21]>$, $<Age \in [22,23]>$, etc., there may be too few people in each group to create any rules. On the other hand, there is a loss of information with large intervals; if the intervals were $<Age \in [0,29]>$, $<Age \in [30,59]>$ and $<Age \in [60,\infty)>$, interesting phenomena for people aged 50-60 could easily be missed. To solve this, Srikant and Agrawal propose an algorithm for automatically partitioning quantitative attributes into suitable number of intervals. Simply put, their idea is to start with assigning one interval to each unique quantitative value and then combine adjacent intervals to create intervals with sufficient amounts of data.

A quite straightforward extension of the Apriori algorithm is to allow attributes to have non-binary values rather than using a series of related binary attributes of which only one can be 1 [36]. While producing the exact same result, this modification gives a large reduction in the number of attributes and consequently speeds up the candidate generation greatly.

3.2.3 Impact rules

Aumann and Lindell considered the algorithm by Srikant and Agrawal inadequate for a number of reasons. In [4], they find that the use of intervals can sometimes be misleading, as illustrated by the following example:

\[
\text{Age} \in [0,14] \Rightarrow \text{Height} \in [100,150] \text{ for 70\% of the population}
\]

and the observation that few 0-1 year olds are 100-150 cm tall. They also criticize the fact that the intervals can always be enlarged, causing an exponential growth of the number of rules, and the loss of information resulting from discretization of real values into intervals.

As an alternative, the authors suggest a new type of rule, based on the distribution of the quantitative values rather than intervals. A rule antecedent then describes a subset of the population, and the consequent consists of one or more quantitative attributes that have an unexpected distribution for the specified subset. Two example rules could be:

\[
\text{Sex} = \text{Female} \Rightarrow \text{Mean wage} = \$7.90/h \text{ (Overall mean wage} = \$9.02)\]

\[
\text{Smoker} = \text{No, Wine drinker} = \text{Yes} \Rightarrow \text{Life expectancy} = 85 \text{ (Overall} = 80)\]

This type of rule has later been named impact rule [34]. To determine whether a quantitative attribute is unexpectedly distributed for some population subset, Aumann and Lindell use the difference between the mean value of the attribute for the subset and the mean for any of its supersets (e.g. the entire population),
but point out that other statistical measures, such as the difference in variance, are possible. To confirm the validity of the rule, they employ a Z-test. The authors note that the abovementioned problems with quantitative association rules are all solved with impact rules. They also state that impact rules are easily understood and interpreted, even when describing complex relations.

The authors suggest using a three stage process to find impact rules. In a first stage, all large itemsets are found, for example using the Apriori algorithm. In a second stage, the mean value of each quantitative attribute is calculated for each large itemset. In a third and most important stage, a lattice structure is built, linking together each itemset with all its super- and subsets. The lattice is then traversed and cases where the mean value between a set and one of its subsets is significantly different are reported as impact rules. The algorithm is more thoroughly described in [4].

In [18], Huang and Webb suggest that impact rules can be found more efficiently by merging these three stages into a single stage. The efficiency gain comes from pruning insignificant sets during the initial search phase, that way it is no longer required to find all the large itemsets, something that according to the authors can be very costly for large and dense databases. They also point out that a t-test is preferred over a Z-test due to potentially small sample sizes.

### 3.2.4 Large and dense data sets

The transaction databases for which the Apriori algorithm was intended are typically of sparse nature. When applying Apriori to dense data, e.g. relational data, the number of itemsets and the resources required to locate them tend to grow exorbitantly. This is a well known problem in the literature. Brin et al. recognize it in a paper from 1997 when they attempt to mine census data with the Apriori algorithm [7]. Out of 127 attributes, they select 73 and convert these to purely categorical attributes. They notice that a number of categories contain over 95% of the population, something that would result in a vast number of large itemsets. For this reason, they remove all categories to which over 80% of the population pertains. They then proceed to mine the data, but have to use a minimum support of at least 36% to limit the number of generated itemsets. With over 20000 generated rules, the majority of which the authors consider not being useful, they conclude that mining dense data such as census data simply is a difficult problem.

In [5] Bayardo et al. suggest that the Apriori algorithm can be made more efficient in mining dense data by incorporating all the constraints that a user has into the search for large itemsets, rather than just the minimum support. Such constraints include minimum confidence in a rule and any other measure a user may have to determine whether a rule is interesting. The paper by Huang and Webb mentioned in section 3.2.3 is based on the same idea and can be seen as an adaption of Aumann and Lindell’s work to dense data.

A frequent problem when mining large or dense data sets with Apriori or one of its siblings, is that all itemsets and itemset candidates can not be fitted into
3.3. OLAP CUBES AND EXCEPTIONS

main memory, requiring data to be repeatedly swapped out to disk [3]. Savasere et al. describe a two-phase algorithm called *Partition* [29] to avoid this problem. First, the original data set is partitioned into small subsets for which Apriori can be run in main memory. An itemset that is large in any of the subsets is said to be a candidate itemset. Second, another pass is made over the data set, verifying which of the candidate itemsets are large in the entire set. Two minimum support thresholds are used, one local for the subsets and one global for the entire set. If the global minimum support is $m$ and there are $p$ partitions, the local minimum support is defined as $\lceil m/p \rceil$. It follows that all large itemsets are guaranteed to be found since an itemset cannot be large in the entire set unless it is large in at least one subset.

3.3 OLAP cubes and exceptions

*OLAP* (On Line Analytical Processing) is a technology that enables users to interactively and effectively examine and summarize data. It is commonly achieved by using a data structure known as an OLAP cube, where data is arranged in a multi-dimensional array. Please refer to [13] for an overview of OLAP and its use in data mining.

Under the supervision of data mining authority Jiawei Han, Qing Chen wrote a thesis concerning the search for exceptions in OLAP cubes [9]. Her work is focused on guided exploration, with visually enhanced tables and box plots aiding the user in his search for exceptional phenomena. While our idea is for the system to do most of the work of such a search and not the user, her thesis does include a thorough and highly relevant discussion of what constitutes an exception.

Several models for measuring exceptions are presented, and a small example will be used in order to look closer at one of them. Let $A$ and $B$ be two categorical attributes by which customers are grouped and where each attribute can take either the value 1 or 2. For each customer $i$, the categorical status and a response value $x_i \in \mathbb{R}$ is known. This data can be presented in a table as follows, with one cell for each combination of values of $A$ and $B$, showing the number of customers $n$ belonging to that combination and their mean response value $m$. All results are rounded to two decimal places. The problem is to decide whether any of the cells contain exceptional (unexpected) values.

<table>
<thead>
<tr>
<th></th>
<th>$A = 1$</th>
<th>$A = 2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B = 1$</td>
<td>$m = 2.33$ ($n=3$)</td>
<td>$m = 7.17$ ($n=12$)</td>
</tr>
<tr>
<td>$B = 2$</td>
<td>$m = 5.86$ ($n=7$)</td>
<td>$m = 9.00$ ($n=18$)</td>
</tr>
</tbody>
</table>

One model mentioned by Chen is the *log-linear model*, which is commonly used to analyze the relationship between categorical variables in a contingency table. It is concluded that this model is suitable for finding exceptions regarding either counts of members or the sum of their responses (which can be transformed into counts),
CHAPTER 3. RELATED WORK

but not for exceptions regarding mean values (which depend on both the number of responses and their sum).

To deal with mean values, Chen describes a *combined effects model*, in which an expected mean and an unexpectedness value is calculated for each cell. The expected value $\mu^G$ for a cell $G$ is calculated as a sum of the contributions $\lambda^H$ of all the categories $H \subset G$ that the cell belongs to. The contribution $\lambda^G$ of a cell $G$ is equal to its measure of unexpectedness and is computed as the difference of the cell’s actual mean value $\bar{x}^G$ and its expected value $\mu^G$. This definition of unexpectedness corresponds to the main effects model described in section 2.2.2 where all categories are assumed to be independent of each other (no interactions exist), but the counts of each cell are taken into consideration. The following formula is obtained:

$$\begin{align*}
\lambda^0 &= \bar{x} \\
\mu^G &= \sum_{H \subset G} \lambda^H \\
\lambda^G &= \bar{x}^G - \mu^G
\end{align*}$$

The global mean value is $\bar{x}$ and the mean of the cell $G$ is denoted $\bar{x}^G$. To exemplify, given the cell $G$: $\{A = 1, B = 1\}$, the following steps are required to compute its expected mean $\mu^{\{A=1,B=1\}}$ and unexpectedness $\lambda^{\{A=1,B=1\}}$:

$$\begin{align*}
\lambda^0 &= \bar{x} = 7.40 \\
\mu^{\{A=1\}} &= \lambda^0 = 7.40 \\
\lambda^{\{A=1\}} &= \bar{x}^{\{A=1\}} - \mu^{\{A=1\}} = 4.80 - 7.40 = -2.60 \\
\mu^{\{B=1\}} &= \lambda^0 = 7.40 \\
\lambda^{\{B=1\}} &= \bar{x}^{\{B=1\}} - \mu^{\{B=1\}} = 6.20 - 7.40 = -1.20 \\
\mu^{\{A=1,B=1\}} &= \lambda^{\{A=1\}} + \lambda^{\{B=1\}} + \lambda^0 = 3.60 \\
\lambda^{\{A=1,B=1\}} &= \bar{x}^{\{A=1,B=1\}} - \mu^{\{A=1,B=1\}} = 2.33 - 3.60 = -1.27
\end{align*}$$

Repeating this procedure for each cell in the table above, the following expected mean values $\mu$ and measures of unexpectedness $\lambda$ are obtained:

<table>
<thead>
<tr>
<th></th>
<th>$A = 1$</th>
<th>$A = 2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B = 1$</td>
<td>$\mu = 3.60 \ (\lambda = -1.27)$</td>
<td>$\mu = 7.07 \ (\lambda = +0.10)$</td>
</tr>
<tr>
<td>$B = 2$</td>
<td>$\mu = 5.52 \ (\lambda = +0.34)$</td>
<td>$\mu = 8.99 \ (\lambda = +0.01)$</td>
</tr>
</tbody>
</table>

Chen states that out of the models she describes, the combined effects model is the one that, in most cases, most accurately describes real-world phenomena. However, she argues that the model becomes too complicated to use when the number of dimensions is high, and hence resorts to other models.
3.4 Rule caching

In [16], Hipp et al. establish that data mining is a highly interactive process, and even though significant progress has been made regarding algorithm efficiency, it is a process that rarely offers true interactivity (at least not for association rule mining). Some researchers have approached this problem by introducing constraints in the mining queries, which can be used during the generation of rules to reduce the number of candidates and speed up the mining process, c.f. subsection 3.2.4. The authors conclude that this is a futile attempt to achieve true interactivity, and instead choose to go in the opposite direction: they remove all the constraints and make the mining query as broad and general as possible. The resulting query will likely execute slowly and generate a vast number of rules, many of which are trivial, uninteresting or spurious. But instead of presenting the rules to the user, they are stored in a rule cache, i.e. a rule database. Subsequent queries can then be answered in real time by retrieving the appropriate rules from the cache instead of running the data mining from scratch.

3.5 Interestingness

A large amount of research has been devoted to trying to assess whether a generated rule is interesting or not. The majority of this research deals with creating various statistical definitions of interestingness, just as has been discussed in the previous sections. Nevertheless, what is statistically interesting is not always interesting to the user. He has other criteria by which he judges what is interesting: most notably if the information is previously unknown to him, useful and something that can be acted upon. Not only would such a user-centered measure of interestingness be difficult to implement; since it is subject to the user’s preferences, it would in fact be impossible to create an objective model suiting all users.

A number of researchers, such as [31, 28, 19], have tackled this by allowing users to set up a model expressing their prior beliefs and knowledge, which can be used to
more accurately determine whether a rule is unexpected or not. However, this can be difficult to set up for real applications, and furthermore, the idea of having to set up a model in advance is incongruous with our vision of a general and automatic system.

In [35], Xin et al. recognize the problem of subjectivity and suggest a solution based on a model adjusting itself according to the user’s interactive feedback. The parameters of the model are derived by iteratively showing a small set of \( k \) sample rules (e.g. \( k = 10 \)) to the user and asking him to rank them according to his interest. This procedure is repeated for several rounds, after which the top-ranked rules are presented to the user. They identify that such a solution consists of two basic research questions: developing a suitable model (as mentioned above) and minimizing the amount of user input required to train the model. The latter problem has previously been studied in [30]. The main novelty of [35] is that their focus is on patterns in data (e.g. association rules) rather than documents (e.g. web pages), the latter being more commonly seen in the literature.

Xin et al. describe two models, the \textit{log-linear model} and the \textit{biased belief model}. For both models, the expected interestingness of a rule is calculated with the help of a weight vector of unknown variables. The user’s rankings are converted to a number of constraints over these variables, resulting in an optimization problem, upon which a support vector machine is used to find a good solution. In the first model, weights are assigned to each item and each interaction between items. The authors are simplifying the model by assuming that all items are independent of each other, ending up with a weight vector of \( n + 1 \) variables for the \( n \) items. The second model is slightly more versatile; instead of using items as a basis, the underlying data (e.g. transactions) is used, with the idea that if a user has high belief in a set of data, he will have high expectations in rules supported by that data. Thus, a belief probability is assigned to each of the \( m \) instances, resulting in a weight vector of \( m \) variables.

In order to minimize the user’s input, the essential idea is to find the rules that allow the system to learn as much as possible, and ask the user to rank those. Inspired by [30], Xin et al. propose that since similar rules naturally rank close to each other, more can be learned from enquiring about the ranking of rules being different from each other. The suggested method is to cluster the rules into \( k \) clusters and select the centroid of each cluster as a rule to be shown to the user. For the first feedback iteration, when the system still has no idea about the interestingness of any rule, all of the \( N \) rules are included in the clustering. For each of the following iterations, and given that it is more important to learn the relative ranking of interesting rules than of uninteresting rules, the \((1 - a)N\) rules ranked as least interesting are removed, and \( N \) is updated as \( N = aN \). The number \( a \) (\( 0 < a < 1 \)) is called the shrinking ratio and this procedure is given the name \textit{progressive shrinking}. The distance measure used for clustering is the Jaccard
3.5. INTERESTINGNESS

distance. Given rules $R_1$ and $R_2$, the distance between them is defined as:

$$D(R_1, R_2) = 1 - \frac{|T(R_1) \cap T(R_2)|}{|T(R_1) \cup T(R_2)|}$$

where $T(R)$ is the set of instances supporting the rule $R$. The authors point out that this is a suitable distance measure for rules mined from transaction databases.
Chapter 4

Mining survey data

This chapter outlines a solution that we believe is effective in mining survey data. Its development was initiated with an orientation phase and a literature study, followed by an iterative process of theoretical reasoning, experimenting and further literature studies. A large number of preliminary experiments have been performed, by creating simple prototypes or using existing tools to test ideas against actual survey data. Some of these experiments will be mentioned briefly throughout the text, but they will not be described in more detail. They merely guided us in the design of a solution that we believe works well and that is further evaluated in chapter 6.

4.1 Method selection

As stated in the introduction, the purpose of this thesis is to find interesting trends and relations out of a large set of data. It is not specified which type of relations to look for, rather it is the point of the mining system to find that out. This is a typical undirected data mining task.

After an extensive review of methods and work related to our problem, it was concluded that the most appropriate choice of data mining technique is either association rules or clustering. Other methods seem to fall into one or several of the following categories:

- Requiring a high degree of user involvement
- Requiring that the user knows beforehand what he is looking for
- Yielding results that are difficult to interpret

Preliminary experiments showed that association rules was a better candidate than clustering. The motivation for this choice is multifold. One reason is that, as mentioned in section 3.1, clustering works best when the number of dimensions is small. This is inharmonious with our problem where we may have over 100 dimensions (corresponding to survey questions). Admittedly, there are techniques for dimen-
tionality reduction: mapping the large input space to a fewer number of dimensions (e.g. principal components analysis) or using only a subset of the dimensions. The first approach may be successful in picking up general patterns in the data, but it seems little likely that a remapped data set would allow the detection of specific and highly localized patterns, such as the hair dryer in a single hotel room being out of order [1].

As for using subsets of the dimensions, section 3.1 described how Luan used a subset consisting of three dimensions that he selected by hand out of the data [21]. Nevertheless, asking the user to select dimensions himself violates our idea of a system that operates mostly without user input. In that aspect, the method described by Agrawal et al. in [1] that automatically finds subspace clusters emerges as a viable alternative.

Another reason for choosing association rules is that we believe that a rule is easier to interpret and base decisions upon than a cluster. Luan uses graphical aids to visualize the demographic attributes of clusters. Experiments with survey data have shown that this approach, while being able to provide some insight into customer behavior, requires a large amount of effort and time by an analyst for results to be analyzed. The method by Agrawal et al. generates cluster descriptions in the form of DNF (disjunctive normal form) expressions. The authors give the following example of a cluster:

\begin{align*}
((30 \leq age < 50) \land (40000 \leq \text{salary} < 80000)) \lor \\
((40 \leq age < 60) \land (20000 \leq \text{salary} < 60000))
\end{align*}

Comparing this type of result to impact rules, as described in section 3.2.3, we believe that impact rules are more easily understood and provide more useful information. Hence, association rules was settled for as the basic technique for mining survey data.

4.2 Data

4.2.1 Available data

The available data is the most crucial part in a data mining project, and the amount and quality thereof directly govern the quality of the final results. In this thesis we have presupposed the existence of a collection of satisfaction surveys containing demographic, behavioral, attitudinal and free text data.

Demographic attributes may include age, income and guest origin. Behavioral data is what a facility records in connection with a guest’s visit. In a hotel setting, it may include which hotel the guest stayed at, the check-in and check-out dates, the room number, the price paid for the room, whether the guest had breakfast at the hotel, etc. Attitudinal attributes contain the customer satisfaction for various factors, e.g. the room, hospitality, bed and food, and are in this thesis assumed to be of Likert type. Attitudinal data may also describe whether the customer had any problems and, if so, in which areas problems were encountered. That type of
4.2. DATA

data is allowed to be dichotomous. Finally, free text data contains guest comments; any problems encountered, ideas for improvements, etc.

In this thesis we will treat Likert type data as interval data. As mentioned in section 2.2.1, this is a common view in practice. If dichotomous data is present, it too will be treated as interval data. There may be better approaches to handle dichotomous data, but the focus in this thesis is on Likert type satisfaction questions.

In line with the discussion in section 2.2.2, demographic and behavioral data will be treated as (weak) independent variables and the attitudinal data will be treated as dependent variables. It is presumed that the business does not have any influence over a guest’s demographics and behavior. Rather, they are seen as preconditions under which it is the job of the business to maximize the satisfaction of the guests.

4.2.2 Preparation of data

The first step in mining data is to assemble all the input data and transform it into a form that can be used by the mining system. For the method we suggest in this chapter, this includes the following steps (not necessarily in the order written):

- Make the raw input data available for the system
- Remove irrelevant data
- Create custom data fields and perform any data transformations
- Convert the data into appropriate data structures

An important point of this thesis is to minimize the amount of manual preparations required by an analyst to set up the data mining system for a new project, and once a project is set up, no manual work should be required in order to keep the system up-to-date. Consequently, the method described here requires very little work by the analyst. A few simple steps of preparation are nevertheless strongly recommended for each new project in order to improve the final results. Compared to the process of designing a new survey and any other non-data mining tasks associated with setting up a new project, we believe that these steps should not be too burdensome. The steps are as follows.

First, the survey questions need to be grouped into three categories: (1) demographic and behavioral attributes, (2) attitudinal attributes and (3) free text data. Refer to section 4.2.1 for a description of each of these categories. For some databases, this step may already have been done, or it may be easy to analyze the questions and automatically detect which category they belong to. For others, there may be ambiguities and a need to manually classify some questions in order to ensure their correct categorization.

Second, and the part that requires the most attention and thought from the analyst, is the removal of any irrelevant attributes. Including every piece of available data in the mining process will make it execute slower and produce results that are on average less useful. For each attribute, the analyst is therefore advised to consider if it could possibly form a part of an interesting pattern. Other criteria may also
CHAPTER 4. MINING SURVEY DATA

<table>
<thead>
<tr>
<th>Technical data</th>
<th>Unnecessarily detailed data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer ID number</td>
<td>Full address (state may be enough)</td>
</tr>
<tr>
<td>Version numbers</td>
<td>Check-in second/millisecond (day or hour may be enough)</td>
</tr>
<tr>
<td>Irrelevant or private data</td>
<td>Any duplicate fields</td>
</tr>
<tr>
<td>Customer name</td>
<td></td>
</tr>
<tr>
<td>E-mail address</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1. Examples of attributes, in a hotel setting, that an analyst may choose to exclude from the mining.

be used in excluding attributes, such as privacy issues. Table 4.1 shows an example of attributes that an analyst may choose to remove. During the above process, the analyst should also pay attention to whether any attribute ought not to be analyzed independently but only in combination with another attribute, and specify any such dependencies. An example is the attribute for which room number a hotel guest stayed in, which makes little sense unless it occurs together with the attribute for which hotel the guest stayed at.

The next step is to define any custom attributes. Such an attribute is created from the existing data with the purpose of allowing specific types of patterns to be found. For example, if the attribute Check-in date exists, one may want to create the custom attribute Check-in weekday in order to search for patterns concerning specific weekdays. See section 4.4.1 for more examples of custom attributes.

The final step is the partitioning and discretization of certain numerical attributes. Being considered as independent variables, all demographic or behavioral attributes will be used as categorical data. This may create a problem if these attributes are of quantitative form, e.g. a customer’s age or the price paid for a hotel room. For demographic data, it may already have been partitioned in the survey – for example a survey respondent may have been asked to select which of a few specified age groups he pertains to. For behavioral data this is more rarely the case. For a discussion of why this may cause problems and how it can be solved algorithmically, see section 3.2.2. In this thesis, we have not experimented with algorithmical solutions, so the method described here requires that the analyst manually defines suitable intervals for those attributes he feels need to be partitioned.

### 4.2.3 Missing data

If survey respondents are allowed to skip survey questions, those questions will result in missing data. As was pointed out in section 2.2.3, this situation can be handled in several ways. If only a few of the surveys are incomplete, the best solution is probably to remove those surveys, leaving only complete surveys to be mined. If a large part of the surveys are incomplete, we believe that the easiest and most
appropriate solution is to include responses on an individual question basis. For example, if we are looking at a customer group consisting of 100 respondents, and we wish to analyze the survey question *Overall satisfaction*, we may find that 99 guests responded to this question. Looking at the question *Satisfaction with the tanning room*, we may find that only 8 guests responded. We do not believe that trying to infer missing values would be helpful, for example trying to infer what the 92 guests that may not even have visited the tanning room would have answered to that question. Instead, we use the results of these 8 guests only, and keep in mind that they are likely to not be representative for the entire customer group of 100 guests, they could for example contain a larger percentage of women or have a lower mean age.

### 4.3 Quantitative association rules

Aumann and Lindell’s impact rules in [4] were invented to solve a problem very similar to ours, and have as such been considered a good starting point for mining survey data. Our suggestion is to use a three stage process just as was done in their paper. In the first stage – finding large itemsets – we suggest using the Apriori algorithm and present a few ideas of how to handle the case when the minimum support level is as low as 2 occurrences, c.f. section 4.3.1. The second stage of computing mean values for each itemset is quite straightforward and should not present any difficulties. For the third stage of actually finding impact rules, a few areas of improvement will be presented in sections 4.3.2 - 4.3.3. Just as in [4], the third stage is intended to be run separately for each quantitative attribute. In a posterior phase, rules concerning the same customer group can be grouped together for improved comprehension of the results. Note that rules grouped together this way are still counted as separate rules in this thesis, not as a single merged rule as in [4].

#### 4.3.1 Finding itemsets

Here, the term itemset corresponds to a customer group and the support of an itemset is defined as the number of customers in that group for whom response data is available. In association rule mining, the point of searching for large itemsets is twofold [2]: first, it ensures statistical significance, and second, small itemsets are not likely to be of business use and are hence not considered interesting. Of course, restricting the search to large itemsets also allows the algorithms to execute faster. In the literature, it is common to see the minimum support allowed to be as low as 0.1% for transaction data [3, 29, 14] and considerably higher for dense data [7, 4].

One of the desires of this project is the ability to find very specific rules, for example that the hair dryer in a single hotel room is out of order. This may perhaps not be of greater business use for an entire chain company, but can still be of interest to the management at a single facility. For such rules to be found, it can be seen that the minimum support must be very low, partly because there may not
be enough data available for the customer group to have high support, and partly because even with enough data, one would want to detect such patterns earlier, not after 20 or 50 customers have reported the problem. Also notice that the argument of statistical significance becomes pointless since we are requiring that each relation passes a significance test to be declared a rule. For these reasons, we will use a minimum support level as low as possible, preferably as low as 2 customers, but memory constraints and large data sets may force this value to be larger.

With potentially millions of survey responses and dozens of categorical attributes, finding every customer group containing more than one customer would be a daunting task, and an important assumption will be made to make that task realizable: we will only look for such small groups within individual facilities. The rationale for this is the one mentioned above, a small group that does not concern a single facility is less likely to be of business use and less likely to be of interest to anyone since managers at individual facilities may feel that rules that do not exclusively concern their facility is not their responsibility [27].

The suggested way to solve the problem is to split the input data into partitions similar to what was described in section 3.2.4, with one partition for each facility. Observe that if there are many facilities, the guaranteed global minimum support will be large. With \( k \) facilities, each analyzed with the minimum support \( m \), itemsets with global support \( \leq k(m - 1) \) are not guaranteed to be found. (Of course, in practice, many itemsets with lower support than that will be found.) This may be acceptable, since the number of customers will increase as the number of facilities does, so even though the global minimum support is large, it may be very small when expressed as a fraction of the total number of customers. If it is not acceptable, and a lower global minimum support is required, a separate pass can be made with facilities grouped into a fewer number of partitions. On an implementational note, it can be seen that the analyses of partitions are independent of each other and the work can easily be parallelized.

We believe that the Apriori algorithm is suitable for finding large itemsets within each facility, in spite of there existing faster methods and the critique of not being able to handle dense data. The reason is that we believe that the first stage of finding large itemsets is rarely the most time consuming part. If it is, an alternative to Apriori should of course be considered.

### 4.3.2 An unexpectedness model

Once all large itemsets have been found, and the statistical measures for them have been collected, they will be used to form impact rules. Just as in [4], a lattice structure is built that links together each itemset with its fathers and sons. The itemset \( X \) is a father to \( Y \) if \( X \subseteq Y \) and \( |Y| = |X| + 1 \). For example, the itemset \( X = \emptyset \) is a father to \( Y = \{ \text{Sex} = \text{Female} \} \) which is a father to \( Z = \{ \text{Sex} = \text{Female}, \text{Age} \in [30,40] \} \). \( X \) is however not a father to \( Z \).

Recall that Aumann and Lindell’s model of an exceptional group is the so called separate effects model; a group whose mean value differs from those of its fathers.
4.3. QUANTITATIVE ASSOCIATION RULES

We consider this conception inadequate since it completely ignores the existence of main effects (see section 2.2.2), and argue that a more elaborate method is required. In fact, if a group consists of more than one category, and no category is completely dominant, the group mean is in general not expected to equal that of any of its fathers.

Section 3.3 described how Chen recognizes this problem and arrives at the conclusion that a combined effects model would be more accurate. Yet, as she adopts the impact rule framework by Aumann and Lindell, it is interesting to see that she does not attempt to incorporate her refined model into their work. Our suggestion is thus to use the combined effects model as described in section 3.3, in particular formula 3.1, to calculate an expected mean value for a group. However, that value is what would be expected under the assumption that all categories are completely independent of each other. As was mentioned in section 2.2.2, this is not the case when categories interact with each other. One category may dominate another, two categories may express the same thing and lack a combined effect, or there can be a combination of these events. In all of these cases, the combined effects model fails and the correct expected value would be closer to the mean value of the dominant father. The same is true if the observation is subject to the ceiling or floor effect. Since these cases seem to be fairly common in practice, and it is difficult to know how and which categories interact, one way is therefore to consider neither of them unexpected. This could be done by defining an interval of values as expected for a group rather than a single one. The interval would be \([\mu_{\text{min}}, \mu_{\text{max}}]\) where

\[
\begin{align*}
\mu_{\text{act}} &= \text{actual group mean} \\
\mu_{\text{exp}} &= \text{expected mean with combined effects model} \\
\mu_i &= \text{actual mean for father } i \\
\mu_{\text{min}} &= \min(\mu_{\text{exp}}, \mu_1, \ldots, \mu_n) \\
\mu_{\text{max}} &= \max(\mu_{\text{exp}}, \mu_1, \ldots, \mu_n)
\end{align*}
\]

In practice, if \(\mu_{\text{min}} \leq \mu_{\text{act}} \leq \mu_{\text{max}}\), we will say that the group mean is expected. If \(\mu_{\text{act}} < \mu_{\text{min}}\) or \(\mu_{\text{act}} > \mu_{\text{max}}\), we will say that the group mean is unexpected and that \(\mu_{\text{min}}\) and \(\mu_{\text{max}}\), respectively, are the expected means.

We will refer to this model as the *restrictive model* and it can be seen as something in between the separate and the combined effects model. It is important to note that while this model will commit slightly more type II errors (failure to detect existing relations) than the two models it is based on, there is also a guaranteed decrease of type I errors (finding spurious relations), since it is more restrictive than both other models. The net result is that a larger part of the rules that are ultimately presented to the user are likely to be found interesting.

4.3.3 Statistical test

When a customer group has been found whose mean value is unexpected, the next step is to perform a statistical test to verify if the difference observed is statistically
significant. In the impact rules literature, the parametric $Z$- and $t$-tests are mentioned to accomplish this test. A $t$-test would be preferred over a $Z$-test since it can not be assumed that the variance of the population is known, and since the sample size may be small, rendering a $Z$-test inaccurate. However, both the $Z$- and $t$-test require the population to be distributed normally. Due to the central limit theorem, this requirement can be overlooked if the sample size is sufficiently large (usually considered 30 or 40). Unfortunately, this is not of much help for the problem set forth in this thesis, since sample sizes are allowed to be as small as 2 respondents. Furthermore, the distribution of survey data responses is often far from normal. An example distribution for a survey question is shown in figure 4.1. Looking back at

![Example distribution for a survey question](image)

**Figure 4.1.** Example distribution for a survey question. The distribution is strongly skewed to the right. One explanation for the large number of responses for 1 could be that dissatisfied people tend to respond 1 rather than 2 or 3. The solid plot shows the distribution of the mean value of a sample of three respondents chosen at random.

what was described in section 2.1.3, it can be seen that this problem is a prime candidate for resampling. Given a completely known population, regardless of its distribution, and a sample thereof, resampling can be used to calculate the exact $p$-value that the difference in mean values between the sample and the population is due to chance. There are a few issues that need to be dealt with before we can settle for this as the statistical test of choice. The most important is that our null hypothesis is not that the means of two populations are equal, but that the mean of a population equals a given expected value (equalling either the minimum or maximum of the expected interval, whichever is closest). Let us look at that in more detail.
Null hypothesis and distribution shifting

By using the restrictive model described in section 4.3.2, the expected mean for a group will either equal the mean of a father or the mean computed with the combined effects model. In the former case, the null hypothesis will simply state that two population means are equal, and will not present a problem since both populations are completely known. However, in the latter case, a population with the sought mean value is not available, making resampling impossible since we lack a population to sample from.

One way to solve this problem is to try to generate the sought population so that a test for identical populations can be used. The restrictive model only results in a mean value, it does not say anything about what distribution a population with that mean value would have. It is nevertheless likely that the distribution would resemble those of its fathers, for which reason we suggest to generate the sought distribution by shifting the father distributions to obtain the given mean.

Equivalent to shifting a distribution by \( x \) units to obtain some mean, is to simply shift the mean we wish to test by \( -x \) units.

This idea will be illustrated with the help of an example. Let \( U \) denote a set of customers, \( S_A = \{A = 1\} \) and \( S_B = \{B = 1\} \) are two sons of \( U \) and \( S_{AB} = \{A = 1,B = 1\} \) is their intersection and a son to \( S_A \) and \( S_B \). The lattice structure and the mean \( m \) and number of customers \( n \) of each group are given in figure 4.2. The data for this example is the same as in section 3.3, but the notation is slightly different. By using the restrictive model, the expected mean for the group \( S_{AB} \) with fully independent categories is 3.60 (c.f. section 3.3) and the expected interval is \([3.60, 6.20]\), meaning that the mean used for testing is 3.60, and that a population with that mean is not available. The initial null hypothesis is thus that the population of \( S_{AB} \) is identical to a population with unknown distribution and mean 3.60 and that the observed difference \( (3.60 - 2.33) \) is due to chance. To approximate the unknown distribution, the idea is to start with a father to \( S_{AB} \), say \( S_A \), which has the mean 4.80, and shift each individual response down by 1.20 units, resulting in \( S'_A \) with a completely known distribution and the mean value 3.60. The new null hypothesis, which can be assessed by resampling, is that the population of
CHAPTER 4. MINING SURVEY DATA

$S_{AB}$ is identical to the population of $S'_A$, and that the observed difference $(3.60 - 2.33)$ is due to chance. As mentioned above, it is equivalent, and much easier, to shift $S_{AB}$ up by 1.20 units instead. So in practice, the original population of $S_A$ would be resampled and tested versus the mean $2.33 + 1.20 = 3.13$.

A few things should be noted about this method. The first is that responses are discrete values while shift distances generally are real values. The second is that shifting responses may give them values that are outside their normal range (e.g. $> 10$ or $< 1$). Lastly, shifts are applied evenly to all responses whereas maybe it should be more likely for e.g. a 9 to be shifted into a 10 than for a 1 to be shifted into a 2. The motivation to still use shifts is that responses throughout this thesis are treated as interval data, so these issues are all left out of consideration. It is nonetheless acknowledged that they are potential sources of inaccuracy.

Testing against all fathers

In the case that a group has several fathers and an unknown distribution needs to be approximated, it remains to decide which of the fathers should be used as a basis for the approximation. One idea could be to use the father whose mean lies closest to the sought mean, since that father can be thought of as best representing the sought distribution. In the above example, that would be $S_A$ since its mean of 4.80 is closest to the sought mean of 3.60. Another idea could be to use the largest father, since the larger a group is, the more likely it is to accurately describe its underlying population (since it is less likely to suffer from sampling errors, this is also referred to as the law of large numbers). In the above example, this would be $S_B$ since it consists of 15 individuals compared to 10 for $S_A$. Depending on which father is used, the computed $p$-value can vary greatly, which in turn can lead to unstable results: if a single piece of data is added that makes the choice go from using one father to another, the $p$-value can change abruptly. To prevent this phenomenon, we suggest that a group should always be tested against all of its fathers, and the final $p$-value should be chosen as the maximum of all the $p$-values computed.

Resampling by bootstrap or permutation

Following the advice given by Simon, as detailed in section 2.1.3, the bootstrap approach of sampling with replacement would be more appropriate if the population is thought to represent a general (infinite) population and the individual responses are independent of each other. In practice, sampling without replacement assigns lower $p$-values to cases where the population is small. For example, consider a population of 9 individuals with the response values $\{1, 1, 1, 7, 7, 7, 8, 8, 8\}$. The probability of drawing the samples $\{1, 1, 1\}$ without replacement is $C(3, 3)/C(9, 3) = 1/84 \approx 1.19\%$. Now imagine that the population is twice as large but contains the same proportions of responses: $\{1, 1, 1, 1, 1, 7, 7, 7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 8, 8, 8\}$. The probability of drawing the same samples is now $C(6, 3)/C(18, 3) = 20/816 \approx 2.45\%$, or more than twice as large as before. Sampling with replacement on the other
4.3. QUANTITATIVE ASSOCIATION RULES

hand means that the size of the population is not relevant, only its proportions of responses. The probability of obtaining the samples \(\{1, 1, 1\}\) is \((1/3)^3 = 1/27 \approx 3.70\%\). Sampling with and without replacement are equivalent if the population is of infinite size.

In this project, we believe that sampling with replacement is more justified than without replacement since we believe that a population should be seen, rather than a fixed set of responses, as a representation of the probability of obtaining each response value in an infinite population. Both methods of sampling have been implemented and test runs confirm the theoretical reasoning above. The implication of assigning lower \(p\)-values to smaller populations is that “complicated” rules (i.e. combinations of more categories) are considered more unexpected, since the populations of such rules are generally smaller.

**Exact version of bootstrapping**

As was discussed in section 2.1.3, it is possible to calculate an exact \(p\)-value by analyzing every possible way to draw samples from the population and count how many of them have a test statistic equally or more extreme than the one observed. As the population and the number of samples grow, this method becomes increasingly less feasible due to the sheer number of possibilities that have to be analyzed. However, if the data can only assume a finite set of integer values, as is the case with Likert scale data, and the test statistic is the mean of the sample, it is possible to efficiently perform the calculation exactly even for large sets of data. We are doing this with the help of a generating function \(G\), defined as:

\[
G(x) = (q_1 x^j + q_2 x^{j+1} + ... + q_k x^{j+k-1})^n
\]

Here, \(j\) is the smallest value assumed by the data and \(j + k - 1\) is the largest. (E.g., \(j = 1\) and \(k = 10\).) \(q_1\) is the probability that the data assumes the value \(j\), \(q_2\) is the probability of the value \(j + 1\), etc., meaning that \(\sum_{i=1}^{k} q_i = 1\). \(n\) is the sample size. Multiplying out, we obtain

\[
G(x) = r_1 x^{nj} + r_2 x^{nj+1} + ... + r_{n(k-1)+1} x^{nj+n(k-1)}
\]

where each coefficient \(r_i\) equals the probability of obtaining the mean value \(j + (i - 1)/n\). The equations for calculating the sought \(p\)-values can therefore be written

\[
p(\text{sample mean} \leq m) = \sum_{i=1}^{[n(m-j)]+1} r_i
\]

\[
p(\text{sample mean} \geq m) = \sum_{i=[n(m-j)]+1}^{n(k-1)+1} r_i
\]

If performed with binary exponentiation, these calculations can easily be done in \(O(kn^2)\) time. A large speedup can be obtained by not using leading and trailing
coefficients that are smaller than $\varepsilon$ for some small $\varepsilon$ in the calculations. In practice, this cuts the run time down to $O(k^2 n)$. Another slight improvement is to compute only the coefficients $r_i$ that will eventually be summed. As can be seen, the run time of the test is proportional to the sample size, which makes it a feasible choice even for large values of $n$. Nonetheless, a faster approach for large $n$ would be to first find an approximate $p$-value with a much quicker parametric test, e.g. a $t$-test, and only perform the exact test if the observation is potentially significant. Such an approach is especially recommended if $k > 10$. If $k$ is large enough for the test to become slow even for small values of $n$, say $n < 40$, the Monte Carlo version of bootstrapping may be a better option, c.f. section 2.1.3.

4.4 Trends

In this section an efficient method to find trend relations is described, where a trend in this context is considered a time-related increase or decrease in satisfaction. For example, a trend relation can state that during the last 30 days, the overall experience at some hotel has a mean value far below of what is expected.

4.4.1 Trends as quantitative association rules

A number of time-related issues can be found by setting up appropriate categories for them. Some examples of useful categories in a hotel environment are Check-in weekday = \{Sunday, ..., Saturday\}, Check-out weekday = \{Sunday, ..., Saturday\}, Weekend = \{No, Yes\}, Check-in monthday = \{1, ..., 31\}. Patterns that occur every other week can be detected with the category Week modulo 2 = \{0, 1\} and similar for every third, fourth, etc. week.

In the rest of this section, we will look into relations concerning a contiguous time period, and the word ‘trend’ will be used to refer to this type of relation. First, some thought should be given as to what is interesting to the user. In this project, it is not considered interesting to detect trends concerning a time period that has already passed, for instance that the customer satisfaction dropped during November last year. Since the system is planned to run constantly, such a trend would have already been discovered a long time ago. Hence, it is considered enough to search for trends concerning data from some point in time up until the most recent data.

By adding a Trend category that can take any integer value $\geq 0$, and with Trend = $d$ signifying the category of guests that have checked out within the last $d$ days, it is seen that this kind of trend can be found by the method described in section 4.3. But there is a problem with this idea; if a precision of a single day is used, a respondent that checked out $d$ days ago would pertain to all the $d + 1$ categories from Trend = 0 up to Trend = $d$. This is not quite congruent with the representation used for survey responses, where only a single value is allowed for each category. Whereas this can be accommodated for in the Apriori algorithm, there is a more serious problem of an explosion of the number of itemsets that are
4.4. TRENDS

generated. For every non-trend itemset, each of the $n$ subsets created by attaching
the attribute $\text{Trend} = d$ for $d = \{0, \ldots, n - 1\}$ must be considered, where $n$ is the
maximum number of days that it is desired to find trends for, for example 90 or
180. This is a hard blow to an already time consuming process.

4.4.2 Trend interestingness

Before starting to search for a more efficient method, let us take another look
at which trends are interesting and what should be reported to the user. Let
us say that, as in the example above, it is found that the satisfaction for some
customer group has been below average during the last $d = 30$ days. It is not
surprising then that the satisfaction has been below average during the last 29,
31, 32, etc., days as well, and presenting all of these relations to the user would
not be particularly useful. A better idea would be to select, using some measure,
the most interesting rule and present only that. Two commonly used measures
of interest are the difference between a group mean and its expected mean, and
the $p$-value describing the statistical significance of that difference. When using
either measure, it is useful to put a threshold level on the other measure in order to
ensure that the results are interesting. In other words, when looking for large mean
differences, it is useful to only consider rules where the $p$-value is below a certain
level of significance, and when looking for small $p$-values, it is useful to only consider
rules where the mean difference is above a certain minimum level. Consequently, the
most interesting trend should be the one starting at either the day that maximizes
the mean difference while being statistically significant, or the day that minimizes
the $p$-value while having a large enough mean difference. In some cases, these days
coincide, but more often the rule with the minimum $p$-value has a larger sample
size, i.e. a larger value of $d$, than the maximum mean difference rule. Figure 4.3
shows an example of this idea.

4.4.3 Finding trends efficiently

Considering the difference between the categories $\text{Trend} = d$ and $\text{Trend} = d + 1$ it
is seen that the latter consists of the same guests as the first, plus the guests that
checked out exactly $d + 1$ days ago. This high degree of similarity suggests that
this kind of trend can be found more efficiently by taking advantage of previous
calculations. The suggested way to do this is to, for each survey question and each
customer group that is to be scanned for trends, start by calculating the mean
value for all the guests that checked out today, then add the data for the guests
that checked out yesterday, then the data for the guests the day before that, etc.
For each day, the expected mean value needs to be calculated, and for that reason
the corresponding mean value of all supergroups is needed. This implies that while
stepping back over time to accumulate data for a group, the corresponding data for
all of its supergroups must also be accumulated. Once the expected mean is known,
it is tested if the mean difference is large enough to be interesting and, if so, if it is statistically significant.

The abovementioned procedure can be made more efficient by first ensuring that all input data is presorted by the check-out date, so that this must not be done for each test. In practice, the majority of groups will be small (since there are more ways to combine categories so to create small groups), and will not contain data from each individual day, in which case it suffices to only deal with the days there actually is data for rather than each single day. Finally, when accumulating data for a group, any computed statistics can be stored so that they can be reused later in case one of its subgroups is being analyzed. If this procedure is still found to perform too slowly, a simple way to speed it up is to consider every 7th or 14th day rather than every single day.

4.4.4 Testing trends statistically

The $p$-value for a trend rule can be calculated the same way as with any other rule by treating the trend attribute as a common categorical attribute. When doing so, one needs to be particularly cautious with the meaning of the $p$-value and when comparing trend $p$-values with non-trend $p$-values, since the algorithm described above selects the interval that is most extreme out of all possible intervals. One
4.5 RULE CACHE

way to deal with this is discussed in section 6.1.3.

4.5 Rule cache

With even a moderate number of surveys, using a rule cache as described in section 3.4 seems to be the only way to allow a user to interact and do mining queries in real time. By updating the rule cache on a frequent basis, such as daily, rules will be up to date and there will be no real drawbacks compared to rerunning every query from scratch. If the rule cache is not too large, the full list of rules can be kept in memory and simply scanned and filtered in real time according to the user query. More sophisticated data structures are required for larger rule caches.

One thing that needs to be taken into consideration is the case where rules are created differently depending on the query parameters. Looking back at the data mining process, it is noted that the only occasion this happens is for the trend rules, so those must be treated somewhat differently.

4.5.1 Trends and the rule cache

Recall that the rule cache is created by running a mining query that is as broad and general as possible. The threshold levels are set to allow a vast amount of rules to be created, after which the user typically runs queries against the cache with more narrowly defined thresholds. This initial use of broadened thresholds will not work very well if only two trend rules are created for each customer group as suggested in section 4.4.2. This is because when the thresholds are more constricted, it may happen that the rule with maximum mean difference no longer is significant enough and the rule with minimum p-value no longer is different enough, whereas in fact there exists a rule somewhere in between which is both different and significant enough.

Preventing this would require storing more trend rules than only the two most extreme ones during the mining process. However, it is not needed to store all the rules; only the ones that could possibly be displayed for some threshold settings. If one rule has both a smaller mean difference and a larger p-value than another rule, it is ensured that it will never be displayed. So the requirement imposed on each stored trend rule is that it must have the largest mean difference of all the rules with a p-value less than or equal to its own. This can be accomplished efficiently during the creation of trend rules by using a sorted linked list structure. The selection of which rule to ultimately display to the user is then left to the scanning/filtering process of the rule cache.

4.6 Presentation

Part of the purpose of this thesis is to produce results that are actionable and easily understood by people without a technical background. This section presents a few
CHAPTER 4. MINING SURVEY DATA

ideas of how that can be accomplished. We suggest that rules are presented in a form such as:

\[
\text{Hotel} = A, \text{Check-in} = \text{Tuesday} \Rightarrow \text{Front desk courteousness} = 6.72 \\
(\text{Expected} = 8.04, \text{Sample size} = 37, \text{Probability} = 0.013\%)
\]

A problem with this presentation is that \(p\)-values may be difficult to interpret for the average user. In the above case, the \(p\)-value 0.013\% equals about \(1/7692\), but depending on the total number of rules that are tested – which even for small sets of data easily exceeds 7692 – it may be no surprise to find rules that significant just by chance (c.f. section 2.1.2). Instead of attempting to explain the concepts of \(p\)-values, null hypotheses and multiple testing to the user, the \(p\)-value can be converted to a strength value \(s\) between 1 and 10, for example using the formula

\[
s = \begin{cases} 
1 & \text{if } p \geq 1\% \\
10 & \text{if } p \leq 0.0001\% \\
1 + \left\lfloor 9 - \log_{10}(p)^{-2} + 0.5 \right\rfloor & \text{otherwise}
\end{cases}
\]

The user could then interpret a rule with a strength of 10 as being almost definitely true, while a strength of 1 means that a rule is likely to be just due to chance. An alternative, simpler, form of the above rule could hence be:

\[
\text{Hotel} = A, \text{Check-in} = \text{Tuesday} \Rightarrow \text{Front desk courteousness} = 6.72 \\
(Difference = -1.32, \text{Strength} = 5)
\]

Another alternative is the following richer description:

The 37 guests at Hotel A that have checked in on Tuesdays average 6.72 on the survey question “Front desk courteousness”, which is 1.32 units below the expected value. The probability that this is due to chance is 0.013\%.

To obtain this specific output, each category and categorical attribute would need to have a description associated with them that fits this form. The last sentence in the output is an incorrect interpretation of what a \(p\)-value is, but may, again, be more successful with non-technical users.

The rules may be presented through an interactive interface in which the user can make queries, sort and filter rules in various ways. Alternatively, the system may select a few of the most unexpected or interesting rules, according to some measure, and show only these.

We believe that a rule can be made more useful by putting it in a wider context. For the example rule above, that could be done by showing the satisfaction for all weekdays, not only the one that generated the rule. That would allow the user to notice any patterns that depend on the weekday. To continue the example, the user may see that the satisfaction is also lower on Wednesdays and Saturdays, and realize that these days coincide with the days a certain front desk clerk works. To assist the user, the satisfaction can be graphed, perhaps using a box plot or some of its variants, instead of displayed numerically.
4.7. SUMMARY

Another idea is to plot the satisfaction for the rule over time. That gives the user an instant overview of when guests visited and how their satisfaction has developed. It enables the detection of whether the rule reflects a pattern that existed in the past or one that still exists. A scatterplot smoother can be used to plot the graph since the satisfaction of individual customers is typically highly irregular.

Finally, a rule can be accompanied by any guest comments and other relevant free text data, which can be used to manually verify a rule. In the example above, guest comments may confirm that the sliding satisfaction is due to a clerk being perceived as rude.

4.7 Summary

This chapter described a powerful, efficient and largely autonomous method for mining survey data consisting of a mixture of categorical and quantitative data. The method is based on what is known in the literature as impact rules, but by taking interaction effects into account and introducing a new restrictive model to counter the uncertainty regarding category dependency, a more accurate definition of what constitutes an unexpected relation was created. It was pointed out that the resulting rules are less prone to type I errors (being spurious).

The flaws in using parametric tests to test for statistical significance were mentioned, and a non-parametric exact test was developed. This test does not require data to be normally distributed, it works with sample sizes as low as a single sample and it has a run time proportional to the sample size.

It was shown how to find a number of trend-related rules, and a method to find rules concerning the last \( d \) days was presented. Finally, a way to interact with the mining system in real time by means of a rule cache was described and some ideas for a user interface were discussed.
Chapter 5

Interestingness

5.1 Subjectivity

So far, our idea of what is interesting to the user has circled around statistical phenomena: customer groups whose mean differs from some statistically expected mean, and \( p \)-values indicating just how unexpected such a difference is. As was discussed in section 3.5, such measures are inadequate for capturing user’s subjective ideas of what is interesting, for which reason our concept of interestingness needs to be further developed. The solution that most closely approaches what we set out to create, i.e. a general system not requiring an excessive amount of manual preliminary input, seems to be a solution based on interactive user feedback. As users are presented with rules by the system, they can be asked to rate each rule for interestingness, feeding the system with information that can be used to learn the users’ preferences. The rest of this chapter describes in more detail how such a system can be assembled.

5.2 A model

5.2.1 Rule components

The first step in creating an interactive system for predicting rule interestingness is to construct a model. For this reason, let us take a quick look at what a rule consists of, and which of these components are influential in its perceived interestingness. The two basic parts of a rule are the customer group and the survey question that the rule concerns. Additionally, the mean difference and \( p \)-value testify to the strength of the rule, and hence can certainly be thought of as having an effect on the interestingness. However, these measures are easy to use in conjunction with thresholds, allowing a user to define a minimum difference and a maximum \( p \)-value for a rule to be considered interesting. That way, we can fully concentrate on the customer group and the survey question.

With the method described in chapter 4, most rules will concern a customer
group within a single facility, meaning that customer groups are generally very small compared to the entire population. For a system to be able to quickly learn to predict the interestingness of rules, it needs to be able to generalize from previous knowledge. A dilemma arises here, because if rules are very specific, how can they be generalized from? For example, a user may think that a certain rule found for hotel A is uninteresting since he knows that hotel A is about to shut down anyway, whereas if the exact same rule were found for the newly opened hotel B it would be highly interesting.

A way to solve this problem is to ask the user to think generally when rating rules, so instead of thinking of the specific categories of a customer group (attributes and values), he should consider the attributes alone. This means not only generalizing rules over facilities, but over all the attributes. For example, if a rule says that rooms on floor 3 at hotel A smells worse than expected, the user should be asked to rate whether rules concerning the smell on an individual floor at a single hotel are generally interesting or not. Even if the user is not interested in this particular rule, he may choose to rate it as interesting in order to see similar rules for other hotels. We see that this allows us to generalize previous knowledge, at the expense of users’ opinions being less specific. The latter can however be cured by giving the user access to simple tools such as hiding individual rules or hiding all rules from a given facility.

5.2.2 The method by Xin et al.

The problem studied by Xin et al. in [35], where rule interestingness was learned by asking users to rank a list of rules, is very similar to our problem, but unfortunately most of their work cannot be applied to our application. The first model for interestingness suggested in this paper, the log-linear model, assumes that all attributes are independent of each other. This results in a model that is too naive for our purposes, since it is highly likely that the interestingness of a rule is affected by how categorical attributes appear together. As an example, consider the attributes Check-in weekday and Check-in monthday. Both of these may be considered interesting on their own, e.g. for finding patterns occurring on Fridays or patterns occurring the first day of every month. However, the combination of both attributes, e.g. patterns occurring when the first day of the month is a Friday, may be considered less interesting.

The second model suggested by Xin et al., the biased belief model, really only works with non-dense data since the interestingness for a rule is based on the underlying data supporting that rule. For survey data, each survey response typically contains all of the attributes, signifying that each rule is supported by almost the entire data set. (At least when rules are represented by attributes and survey questions, and not by the specific categories, c.f. section 5.2.1.)
5.3 MEMORY-BASED REASONING

5.2.3 Other models

For the reasons mentioned in the previous section, we see that any linear classifier, be it a Naive Bayes classifier or a perceptron neural network, will be insufficient for our problem.

Note that the log-linear model can easily be extended to include interactions between two or more attributes. This however means that the number of unknown variables grows markedly and may make it more difficult for the system to predict the interestingness of interactions that it has not been trained for. In [35], no experiments were carried out with log-linear interactions, and neither has been done in this thesis, so its effectiveness in practice for our application is unknown.

Other options for models are non-linear neural networks and support vector machines. As for neural networks, some of the drawbacks associated with them were discussed in section 2.3.3. We consider it a major problem that predictions cannot be motivated properly, since users may find such predictions hard to trust. We also believe that it could be difficult to use a neural network as a general solution, since parameters that work well for one set of data may be less optimal for another. Support vector machines are powerful tools for classification and regression, albeit suffering from a similar problem of parameter selection for the kernel function [8]. They have not been explored further in this thesis.

The model we believe is most appropriate for this project is memory-based reasoning (MBR). There are a number of advantages of using this model, most importantly that it is very versatile; the only adaptation that needs to be done if the underlying task changes is that the distance function between rules changes accordingly. Since a distance function can be constructed for most tasks, MBR is a good choice in order to enhance the generality of our system. MBR is also an efficient model, scaling well as the number of predicted and rated rules grow, and thus suitable to use for real time interaction.

5.3 Memory-based reasoning

This section will treat how memory-based reasoning can be applied to our problem of predicting rule interestingness. A crucial part of MBR is how to measure the distance between rules; this topic is further explored in section 5.4.

5.3.1 Prediction of interestingness

Following the basic ideas of MBR, we compute the predicted interestingness of a queried rule $Q$ as an average of already rated rules. The first step is to sort the set of rated rules by increasing distance. The distance function used here is called $D$ and ranges from 0 to 1, with $D(R_1, R_2) = 0$ signifying that the two rules $R_1$ and $R_2$ are considered identical and $D(R_1, R_2) = 1$ signifying that they are as different from each other as possible. Rules are rated by assigning a value $0 \leq r \leq 1$ to them, with $r = 0$ meaning that a rule is completely uninteresting and $r = 1$
meaning that it is very interesting. The sorting step results in the rule set \( N \) where \( D(Q, N_i) \leq D(Q, N_j) \) and \( 1 \leq i < j \leq m \). Here, \( m \) is the total number of rated rules. If \( D(Q, N_1) = 0 \), it means that at least one rule considered identical to \( Q \) has already been rated, and the interestingness for \( Q \) will then be predicted by a simple unweighted average of the ratings \( r \) over all such rules, resulting in the prediction \( p(Q) \):

\[
p(Q) = \bar{r}(N_i | D(Q, N_i) = 0)
\]

If \( D(Q, N_1) > 0 \), the \( k \) nearest neighbors are used to compute a weighted average. The variable \( k \) can be set to a fixed value, e.g. \( k = 5 \), or it can be given a low value initially when few rules have been rated and increased as the amount of user feedback grows. Of course, \( k \) can never be larger than \( m \). The usage of the \( k \) nearest neighbors rather than the entire set of rated rules is to prevent the rating given to a rule to affect the predictions of very distant rules. The following weight function \( w \) is defined with the purpose of boosting weights for rules \( N_i \) that are close to \( Q \):

\[
w(Q, N_i) = \ln \frac{1}{D(Q, N_i)}
\]

The formula used for predicting the interestingness is:

\[
p(Q) = \frac{\sum_{i=1}^{\min(m,k)} w(Q, N_i) r(N_i) + b/2}{\sum_{i=1}^{\min(m,k)} w(Q, N_i) + b}
\]

where \( b \) is a small weight, e.g. \( b = 0.5 \), used to bias the result towards 0.5, which is thought of as an interestingness level of “neutral”. This is particularly needed when either \( m = 0 \) or there are no neighbors in the proximity of \( Q \).

### 5.3.2 Certainty of prediction

In addition to a predicted interestingness, it is useful to have a measure of how certain one can be of the prediction. Say for example that a rule identical to the queried rule \( Q \) has been given a rating of 0.35. We can then be 100\% certain that the interestingness of \( Q \) is 0.35 as well, and \( Q \) is thus not of any interest.

On the other hand, if there is only one rated rule, with a rating of 0 and a distance of 0.8 from \( Q \), and with \( b = 0.5 \), the predicted interestingness for \( Q \) using formula 5.1 is again 0.35, yet this time we cannot be as certain that the prediction is correct. Since the rated rule is so distant from \( Q \), there is a good chance that the user has a different opinion about \( Q \) than what is predicted, and it may be worth for the user to have a look at \( Q \) in spite of the predicted value being so low.

To aid the user in this decision, a certainty value \( c \) (\( 0 \leq c \leq 1 \)) can be computed and showed along with the prediction. If \( D(Q, N_1) = 0 \), \( c \) is set to 1 (100\% certainty). Otherwise, \( c \) is defined as a weighted average of the similarity to the \( k \) nearest neighbors, where similarity is defined as 1 − distance:

\[
c(Q) = \frac{\sum_{i=1}^{\min(m,k)} w(Q, N_i)(1 - D(Q, N_i))}{\sum_{i=1}^{\min(m,k)} w(Q, N_i)}
\]
This formula is similar to the computation of $p$ above, and the same notations are used.

## 5.4 Rule distance

### 5.4.1 Existing methods

The ability to measure the distance between rules in a sensible way is crucial for many purposes, e.g. clustering or classification tasks. Xin et al. studied this problem in order to cluster rules and settled on the Jaccard distance (c.f. section 3.5). Unfortunately, this measure does not work with our dense data since, as mentioned in section 5.2.2, each rule is supported by almost the entire data set, giving all rules a distance close to 0 from each other. Representing rules by their specific categories instead of just the attributes would not be helpful. Consider for example rule $A$ concerning the customer group $<\text{Hotel} = A, \text{Room number} = 0235>$ and rule $B$ being identical to rule $A$ except concerning another room, say room number 0236. The intersection of the set of survey responses concerned by $A$ and $B$ is guaranteed to be empty, since each response is linked to only one hotel room, making rules $A$ and $B$ as different from each other as possible, which almost certainly is not desired.

Another idea is to base the measure on the attributes rather than on the underlying data, which means that we need to be able to measure the distance between attributes, categorical as well as quantitative. A very naive approach is to just test the attributes for equality, giving two attributes a distance of 0 if they are the same and 1 otherwise. The disadvantage of this approach is of course that two attributes may be intuitively very close to each other, for example the categorical attributes $\text{Check-in weekday}$ and $\text{Check-out weekday}$ or the quantitative attributes $\text{Front desk courteousness}$ and $\text{Front desk helpfulness}$, yet be defined as being maximally different from each other. Preferably, one would want a more fine-grained distance measure than this, and one that tries to resemble the perceived distance better. In what follows, a method will be described of how that can be done automatically without any user input.

### 5.4.2 Distance and similarity

Equivalent to measuring rule distance $D$ is measuring rule similarity $S$ ($0 \leq S \leq 1$) where $D = 1 - S$. To simplify the problem, a rule is split into its components and the similarity between two rules $A$ and $B$ can then be defined as $S(A, B) = f(S_c(A, B), S_q(A, B))$, where $S_c$ measures the similarity between the categorical attributes of $A$ and $B$ and $S_q$ measures the similarity between their quantitative attributes. The function $f$ combines the two similarity measures into a final similarity value, and can be as simple as $f(c, q) = cq$. 

45
5.4.3 Similarity between quantitative attributes

To measure the similarity between two survey questions, we compute how well they correlate. More specifically, pairwise Pearson correlation coefficients are calculated between the survey questions for the entire set of survey responses, and the absolute values (in the interval $[0, 1]$) are used as similarity measures. Note that two questions correlating well does not necessarily imply that they are perceived as being similar. Other factors, such as the ordering of questions in the questionnaire, can play an important role, which hence must be considered a flaw of this method. Still, we believe that correlation is a good method when a user is not available to explicitly define the similarity between all attributes.

5.4.4 Similarity between categorical attributes

For obvious reasons, correlation coefficients can not be used for the categorical attributes as they were for the quantitative ones, and a different method is called for. Looking through the list of generated rules, especially rules with $p$-values close to 0 (thus being less likely to be spurious), it can be observed that categorical attributes that are intuitively similar to each other tend to occur with the same set of survey questions. For example, both rules including the attribute Room number and rules including Floor number tend to occur with survey questions concerning room issues. Rules including the attribute Check-in weekday and rules including Check-out weekday all tend to occur with survey questions that can be linked to trend-related issues.

This insight can be used to build a profile for each categorical attribute, describing how strongly it is connected to each question. We suggest to use a simple form of Hebbian learning [15] to create such a profile. For each of the $m$ rules in the complete list of rules, a pattern vector $x$ with $n$ elements is created, one element for each categorical and quantitative attribute. The vector is defined as:

$$x_{k,i} = \begin{cases} 0 & \text{if } i:th \text{ attribute is not included in rule } k \\ \sqrt{\frac{\ln(p(1-\varepsilon)+\varepsilon)}{\ln \varepsilon}} & \text{if } i:th \text{ attribute is included in rule } k \end{cases}$$

Note that $0 \leq x_{k,i} \leq 1$ as long as $0 \leq p \leq 1$, and $x_{k,i}$ is larger for smaller values of $p$. $\varepsilon$ is a small epsilon value, e.g. $\varepsilon = 10^{-12}$. Next the following formula for Hebbian learning is applied:

$$w_{i,j} = \frac{1}{n} \sum_{k=1}^{m} x_{k,i} x_{k,j}$$

where $w_{i,j}$ is the weight of the connection between attribute $i$ and $j$. The profile mentioned above is obtained for a categorical attribute $i$ by considering those $w_{i,j}$ where $j$ corresponds to a quantitative attribute.

The final problem is how to convert two profiles into a measure of similarity. For this, the connection weights are converted to ranks, and the Kendall $\tau$ rank
correlation coefficient [20] is computed in order to compare the order in which questions are ranked. Any negative coefficients are set to 0, ending up with a similarity measure in the interval $[0, 1]$.

Note that this profile approach can be used to measure the similarity between survey questions as well, however, during preliminary testing, standard Pearson correlation coefficients seemed to yield a more intuitively correct result.

### 5.4.5 Multiple categorical attributes

The situation becomes slightly more complicated when we wish to compare a set of categorical attributes to another, rather than just comparing single attributes. As we will see, in order to yield estimates of rule similarity that is useful for MBR, it becomes necessary to know which of the compared rules is the query rule, $R_q$, and which is the rated rule, $R_r$, and furthermore, we need to know the interestingness rating $t$ that the user gave to $R_r$. Here, $0 \leq t \leq 1$ where 0 is the least interesting and 1 the most. Recall that the estimated interestingness of a queried rule can be seen as a weighted average of the interestingness of the rated rules, where larger weights are given to more similar rules, c.f. section 5.3.1.

The idea for estimating the similarity between two sets of attributes is explained below. Note that the mentioning of a survey question has been left out to simplify the description, so the following section should therefore be thought of as concerning a fixed survey question.

If the set of categorical attributes $C_r$ (of the rule $R_r$) has been rated as being interesting to the user ($t = 1$), then each subset, specifically each individual attribute, of $C_r$ is likely to also be considered interesting. This follows from the reflection that if any attribute or set of attributes is considered uninteresting, it would likely remain uninteresting if more attributes were added. There are exceptions to this idea, but more often it is a correct assumption, and as such, it helps to create more accurate estimates.

Consequently, if an attribute $q$ of the attribute set $C_q$ (of the rule $R_q$) is similar to any of the attributes of $C_r$ (regardless of $q$’s similarity to the other attributes of $C_r$), we would like to say that $q$ is similar to $C_r$ in order to allow $R_r$ to have a larger effect on the interestingness of $R_q$. Hence, for each $q$ we find its most similar attribute in $R$, and at the end we compute the total similarity $s_q$ between $C_q$ and $C_r$ as the product of all of these similarities.

On the contrary, if $C_r$ has been rated as being uninteresting ($t = 0$), it is difficult to say anything about subsets of $C_r$, since it is highly possible that some subsets are in fact interesting, but the combination of all attributes is not. Thus, in order for $C_r$ to be similar to $C_q$ and thereby for $R_r$ to have a larger effect on the interestingness of $R_q$, we need that each attribute $r$ of $C_r$ is similar to any attribute of $C_q$. This situation can be seen as the opposite of the one above when $t$ equalled 1, and accordingly we obtain the total similarity $s_r$ between $C_q$ and $C_r$ as the product of the maximum similarity in $C_q$ for each attribute of $C_r$.

For $0 < t < 1$, we use a weighted average of $s_q$ and $s_r$, resulting in the below
CHAPTER 5. INTERESTINGNESS

<table>
<thead>
<tr>
<th>Room number</th>
<th>Check-in weekday</th>
<th>Week modulo 2</th>
<th>Check-out weekday</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>20%</td>
<td>80%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15%</td>
<td>90%</td>
</tr>
</tbody>
</table>

Table 5.1. Sample similarities between the individual categorical attributes used in figure 5.1.

The formula to calculate the similarity $S_c$ between $C_q$ and $C_r$. Here, $s_{cc}$ gives the similarity between two individual categorical attributes as described in section 5.4.4.

$$s_q = \prod_{q \in C_q} \max(s_{cc}(q, r) | r \in C_r)$$

$$s_r = \prod_{r \in C_r} \max(s_{cc}(q, r) | q \in C_q)$$

$$S_c(C_q, C_r, t) = ts_q + (1 - t)s_r$$

Note that if both $C_q$ and $C_r$ contain a single attribute, $q$ and $r$ respectively, we have that $S_c = s_q = s_r = s_{cc}(q, r)$. A thorough example is given by table 5.1 and figure 5.1 in order to clarify the above method.

![Scenario 1: Room number = A, Check-in weekday = B, t = 1](image1)

Scenario 1: $R_r = A$, $R_q = B$, $t = 1$

Room number | Check-in weekday
-------------|------------------|
80%          | 90%              |

Week modulo 2 | Check-out weekday
-------------|------------------|
20%          | 90%              |

Similarity = 80% * 90% = 72%

![Scenario 2: Room number = A, Check-in weekday = B, t = 0](image2)

Scenario 2: $R_r = A$, $R_q = B$, $t = 0$

Room number | Check-in weekday
-------------|------------------|
80%          | 90%              |

Week modulo 2 | Check-out weekday
-------------|------------------|
20%          | 90%              |

Similarity = 20% * 90% = 18%

![Scenario 3: Room number = B, Check-in weekday = A, t = 1](image3)

Scenario 3: $R_r = B$, $R_q = A$, $t = 1$

Room number | Check-in weekday
-------------|------------------|
80%          | 90%              |

Week modulo 2 | Check-out weekday
-------------|------------------|
20%          | 90%              |

Similarity = 20% * 90% = 18%

![Scenario 4: Room number = B, Check-in weekday = A, t = 0](image4)

Scenario 4: $R_r = B$, $R_q = A$, $t = 0$

Room number | Check-in weekday
-------------|------------------|
80%          | 90%              |

Week modulo 2 | Check-out weekday
-------------|------------------|
20%          | 90%              |

Similarity = 80% * 90% = 72%

Figure 5.1. Four different scenarios for the similarity between rule $A$ concerning the categorical attributes Room number and Check-in weekday and rule $B$ concerning Week modulo 2 and Check-out weekday. In scenario 1, the queried rule $B$ is computed as being 72% similar to the rated rule $A$ since $A$ has been rated as interesting and both attributes of $B$ are highly similar to one of the attributes of $A$. Based only on this, the interestingness of $B$ would be predicted to be quite high. In scenario 2, $B$ is only 18% similar to $A$ since $A$ has been rated as uninteresting and the attribute Room number of $A$ is not very similar to any of the attributes of $B$. Perhaps it is this attribute that makes $A$ uninteresting. This uncertainty is reflected by the low similarity value, which implies that the predicted interestingness of $B$ is hardly affected by $A$. The remaining scenarios, with $A$ and $B$ swapped, can be thought of as opposite versions of the first two scenarios.
Refining the similarity measure

As the user rates more and more rules, it would be possible to refine the similarity measures used between attributes. Say for example that the similarity between two attributes is very high according to the system, yet the user gives them very different ratings. This would be an indicator that the attributes should in fact be more distant from each other. There are some obstacles that must be overcome in embodying this idea; first of all, in practice, attributes are not rated by themselves, but in various combinations, so one needs to take into consideration the interaction between attributes. Furthermore, after determining how ratings relate to similarity, it must be examined how to combine that with the existing similarity measure. For these reasons, this idea has not been analyzed any further in this thesis.

User input

The intended way for the user to interact with the system and leave feedback is to run queries against the rule cache, look at the resulting rules, and rate them according to how interesting he thinks they are. To the system, a rating $t$ is a real value ranged between 0 and 1, but to simplify for the user, any scale can be used. It can be as simple as asking the user to rate rules as being either interesting ($t = 1$) or uninteresting ($t = 0$), or a five-point scale can be used ($t$ being one of $\{0, 0.25, 0.5, 0.75, 1\}$), or perhaps a slider control allowing a wider range of values.

There is no need for the user to rate all the rules that he is presented with. For example, he may choose to rate only those rules for which he disagrees with the predicted rating.

Rating versus ranking rules

Rating rules as compared to asking the user to rank a set of rules (which was studied in [35]) has both advantages and disadvantages. An obvious advantage of rated rules is that the complexity of the problem is greatly reduced, since the ratings are explicitly specified by the user. With sets of ranked rules, we would need to find the optimal global ranking, which, as mentioned in [35], turns into an NP-hard problem. In [35], this was solved by using a ranking support vector machine.

Moreover, rule rating may be seen as an easier task for the user, since to rate a rule he only needs to think about how interesting he considers that rule to be, not how interesting it is compared to a number of other rules. Ratings can thus be collected from a multitude of users rating rules casually, much as how Google Groups allows users to rate Usenet posts and uses those ratings to determine the overall rating of a post or thread.

Rule ranking on the other hand is able to more accurately order rules by a user's interest. Let us say that there are a number of rules that are all considered very interesting. With rule rating, they would typically all be given the rating $t = 1$, and would thus be considered equally interesting by the system. For the system
to order these rules between themselves it would have to use some other measure, e.g. their p-value. It would of course be possible for a user to be more specific in his ratings, such as varying them between 0.95 and 1, but it is unrealistic to assume that users are that meticulous in their ratings. Asking the user to rank the same set of rules would be a much more user friendly way to achieve an accurate ordering by interest.

There is nevertheless one issue with rule ranking that makes the final decision easy, namely that the rankings need to be converted to a set of constraints over the underlying model parameters, which in turn poses restrictions on the model used. Since MBR does not have any model parameters, but use the rated/ranked rules themselves as the model, the constraints would have to be set up over the rules directly. This would however not be very useful, as it would leave many ambiguities when combining sets of rankings. For example, if the rules A and B have been ranked as [A, B], signifying that A is more interesting than B, and the rules C and D have been ranked as [C, D], the combined ranking of all four rules can be any of {[A, B, C, D], [A, C, B, D], [A, C, D, B], [C, A, B, D], [C, A, D, B], [C, D, A, B]}. To solve this ambiguity, information would be needed about how A and B rank against C and D, which means either asking the user to provide that information or trying to solve it by breaking the rules down into their components (categorical and quantitative attributes). The first option is not viable since it would require an unreasonable amount of user input when the number of ranked rules grow, and the second option means going away from MBR as our model. As a result, rule ranking has not been explored further in this thesis. Whether a model exists that is suitable for our application and for which rule ranking works well has not been looked into.

5.5.2 Clustering and optimal learning

Depending on how the system is to be used, the clustering framework used by Xin et al. in [35] and described in section 5.2.2 may be an interesting approach to try to speed up the training process of the model. In our case, we have alternative measures – p-values and mean differences – by which to sort rules if the model has not been trained sufficiently. The training of the model is thus imagined to take place during and as long as the system is used, by asking the user to rate the rules that he is viewing. If there are multiple users, the system can use a separate model for each user, thereby adapting itself to each user’s individual preferences.

If the idea of continuous training is undesired, an initial clustering-based training phase very much like the one described in [35] is believed to be a good alternative. This idea has been experimented with and is discussed further in section 6.4.

5.6 Summary

In this chapter, memory-based reasoning was suggested as a method to learn and to predict the users’ interests in rules. The predicted interest for a rule was calculated
5.6. SUMMARY

as an average of the interest of similar rules, weighted by each rule’s similarity. Much effort was put into finding an appropriate rule distance function, and a fully automatic function was suggested that works by estimating the distance between quantitative and categorical attributes separately and combines the results. The bases for the estimations are the raw response data and the list of generated rules. The suggested way for a user to interact with the system is to rate shown rules between 0 (completely uninteresting) and 1 (very interesting).
Chapter 6

Results

The method described in chapter 4 and 5 has been implemented in Java and evaluated in four aspects: its ability to find exceptional phenomena, its scalability, its aptitude for learning a user’s interests and its feedback model. The test machine used in all experiments was a Dell Precision M65 laptop computer with an Intel Core Duo 2 GHz processor and 2 GB of RAM. Java version 1.6.0 was used.

6.1 Finding exceptional phenomena

6.1.1 Data

The data in this experiment is authentic and comes from a large hotel chain with over 500 hotels and several hundred thousands of survey responses. Out of a total of about 300 attributes, 29 categorical attributes and 68 quantitative attributes were manually selected according to the procedure in section 4.2.2. All quantitative attributes ranged from 1 to 10. Many surveys had a few missing attributes, categorical as well as quantitative. In total, the categorical attributes contained over 6000 different categories, but each individual hotel had much fewer categories than that. To summarize the median number of categories for the categorical attributes within each hotel, 7 attributes had a median of 0-2 categories, 7 attributes had 3-4 categories, 8 had 5-9 categories, 1 had 10-19 categories, 4 had 20-49 categories and 2 had 50-99 categories.

6.1.2 The experiment

To verify that the system is able to find unexpected relations and that those relations are indeed unexpected and not the result of having searched hard enough, the authentic data was mined and a few categorical and quantitative attributes were selected for analysis as listed in tables 6.1 and 6.2. For each combination of categorical and quantitative attributes, the number of rules concerning those attributes were counted. This was repeated for a couple of different significance levels, ranging from 0.01 (1%) down to 0.000001 (one out of a million) and the results are shown in
CHAPTER 6. RESULTS

Table 6.1. Categorical attribute names and descriptions. The maximum and median number of categories for each attribute is listed. Note that the recent trend attribute is treated differently, as described in section 4.4.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Max. #categories</th>
<th>Median #</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Purpose of visit</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Floor</td>
<td>Hotel floor number</td>
<td>1-39</td>
<td>3</td>
</tr>
<tr>
<td>Price</td>
<td>Price paid for room</td>
<td>14</td>
<td>7</td>
</tr>
<tr>
<td>Check-in</td>
<td>Check-in weekday</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Trend</td>
<td>Recent trend</td>
<td>200</td>
<td>98</td>
</tr>
</tbody>
</table>

Table 6.2. Quantitative attribute names and descriptions. All questions are 10 point Likert scale questions.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experience</td>
<td>Overall experience with stay</td>
</tr>
<tr>
<td>Value</td>
<td>Value for money</td>
</tr>
<tr>
<td>Room</td>
<td>Overall satisfaction with room</td>
</tr>
<tr>
<td>Arrival</td>
<td>Speed of arrival</td>
</tr>
</tbody>
</table>

Table 6.3. The number of rules for a selection of attributes for the authentic data. The significance levels used are $p_1 = 10^{-2}$, $p_2 = 10^{-3}$, $p_3 = 10^{-4}$, and $p_4 = 10^{-6}$.

<table>
<thead>
<tr>
<th></th>
<th>Experience</th>
<th>Value</th>
<th>Room</th>
<th>Arrival</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$p_1$</td>
<td>$p_2$</td>
<td>$p_3$</td>
<td>$p_4$</td>
</tr>
<tr>
<td>Purpose</td>
<td>12</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Floor</td>
<td>98</td>
<td>29</td>
<td>12</td>
<td>5</td>
</tr>
<tr>
<td>Price</td>
<td>42</td>
<td>13</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Check-in</td>
<td>35</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Trend</td>
<td>107</td>
<td>45</td>
<td>28</td>
<td>11</td>
</tr>
<tr>
<td>Sum</td>
<td>1338</td>
<td>447</td>
<td>209</td>
<td>81</td>
</tr>
</tbody>
</table>

Next, the responses to each quantitative attribute were randomly shuffled within each hotel, producing a scenario with the same proportions of response values for each question, but where any relations between categorical and quantitative attributes ought to have vanished. The exact same procedure as with the real data was repeated with the permuted data, and the results are shown in table 6.4.

6.1.3 Results

The results from table 6.3 show that there are many strong rules connecting the floor number and the room satisfaction, and likewise the price paid and the perceived
6.1. FINDING EXCEPTIONAL PHENOMENA

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Floor</th>
<th>Price</th>
<th>Check-in</th>
<th>Trend</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>p2</td>
<td>p3</td>
<td>p4</td>
<td>p1</td>
<td>p2</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td>11</td>
<td>10</td>
<td>18</td>
<td>230</td>
</tr>
</tbody>
</table>

Table 6.4. The number of rules for a selection of attributes for the randomly permuted data. The significance levels used are \( p_1 = 10^{-2} \), \( p_2 = 10^{-3} \), \( p_3 = 10^{-4} \), and \( p_4 = 10^{-6} \).

value for money. The trend attribute is strongly connected to all questions, but mostly with the overall experience and the value for money. The check-in weekday gives rise to two extremely significant \( (p < 10^{-6}) \) rules concerning the speed of arrival, whereas neither Purpose, Floor or Check-in offer any greater surprises for that question.

Keep in mind that rules denote unexpected phenomena. For example, if there is a general pattern stating that discounted rooms lead to better value for money, it would show up as a single general rule, and thenceforth be expected for each hotel. Consequently, any other rule concerning Price and Value would have to express a difference from the general case; in this case it could perhaps be an indicator of hotels that have priced their rooms too low or too high.

The difference between table 6.3 and 6.4 is striking and verifies that a large part of the results in table 6.3 are truly unexpected. With randomly permuted data, not a single rule was found extremely significant and only one of the analyzed rules had a \( p \)-value less than \( 10^{-4} \). Note that the trend category generates the most number of rules, and is also the category where the most significant rule is found in table 6.4. This could be expected due to the reasons mentioned in section 4.4.4.

As discussed in 2.1.2, \( p \)-values should be read with caution since it is expected to find low \( p \)-values just by chance. With real data, the statistical tests performed are typically highly dependent of each other since many attributes correlate strongly, and it becomes very difficult to use corrections such as the Bonferroni method. Neither do we know how an end user will use the system and its output; perhaps he has a clearly defined hypothesis that he wishes to test, and in that case it would have been wrong to have “corrected” the significance level. Our belief is therefore that it is best to leave the \( p \)-values as they are, but inform the user that they should be interpreted with care. To give the user an idea of what \( p \)-values he should be expecting to find by happenstance, it is suggested to add one or a few random attributes and give the user the choice to include these random attributes in the output, that way observing how significant random rules are considered, much like we did in this experiment.
CHAPTER 6. RESULTS

<table>
<thead>
<tr>
<th>Description</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of facilities</td>
<td>50</td>
</tr>
<tr>
<td>Number of customers per facility</td>
<td>500</td>
</tr>
<tr>
<td>Number of categorical attributes</td>
<td>20</td>
</tr>
<tr>
<td>Number of quantitative attributes</td>
<td>50</td>
</tr>
<tr>
<td>Number of levels per quantitative attribute</td>
<td>10</td>
</tr>
<tr>
<td>Minimum $p$-value for being significant</td>
<td>0.01</td>
</tr>
<tr>
<td>Minimum mean difference for being interesting</td>
<td>0.25</td>
</tr>
<tr>
<td>Minimum customer group size</td>
<td>2</td>
</tr>
<tr>
<td>Maximum number of attributes in a customer group (rule complexity)</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 6.5. The parameters that were varied during the tests, and their default values.

6.2 Scalability

6.2.1 The experiment

To assess the performance of the mining process, a large number of tests have been run on synthetically created random data with varied input parameters. Those parameters, and their default values used during the tests are shown in table 6.5. As a comparison, the experiments with authentic data from the previous experiment were also timed. The authentic data was mined with the default settings for significance, mean difference and customer group size as shown in table 6.5. Recent trend rules were included in the mining. For the synthetic data, all attributes were assigned uniformly random values/categories. The quantitative attributes ranged from 1-10, discretized in the number of steps given by the number of quantitative levels. The motivation for using random data rather than actual data is mainly to isolate the effect of each individual parameter and not need to worry about properties of the data distorting the measurements. As will be seen by the results from the experiment with authentic data, the run time is in any case not much dependent on which patterns exist in data, since most significance tests need to be performed anyway to determine the significance of a relation. However, as will also be seen, the run time is very dependent on the number of categorical attributes and the number of categories per categorical attribute, since this affects how many customer groups are created in the very first step of the mining procedure. To make the tests more authentical in this aspect, the number of categories were varied so that each test had an equal number of attributes having 2, 3, 7, 12 and 100 categories. (In other words, all tests had $5n$ categorical attributes for some $n \in \mathbb{N}$.) These numbers were chosen since they are believed to be a good representation of attributes that may exist in real data. 2 and 3 categories may represent binary attributes such as gender or attributes that have several categories but where most customers end up
6.2. SCALABILITY

in only a few of them. 7 and 12 categories are thought to represent demographical attributes such as age group and occupation or trend attributes such as weekdays. 100 categories are thought to represent specific attributes such as room number or customer origin.

Figure 6.1 shows the elapsed time and number of rules generated by varying each of the parameters in table 6.5, while all other parameters assumed their default values and trend rules were not looked for. There is also one series of tests with fixed numbers of categories rather than varied.

6.2.2 Results for authentic data

The time elapsed for the mining of the authentic data was 45 minutes and a total of 174161 rules were found. As a comparison, the randomly permuted data also took 45 minutes to mine and 133605 rules were generated. Thus it would easily be possible to perform the mining once a day to keep the rule database constantly up-to-date.

Out of these 45 minutes, only 43 seconds were spent by the Apriori algorithm. 15 minutes were consumed computing mean values and collecting statistics (stage two of the three stage mining process described in section 4.3) and 29 minutes were spent actually finding rules; mainly parsing the itemset graph, computing \( p \)-values and searching for trend rules.

56% of the generated rules had only a single categorical attribute in the rule antecedent and the remaining 44% had two attributes. If allowing a rule to have three categorical attributes (apart from the hotel attribute and the recent trend attribute, which are treated differently), the Apriori algorithm stopped due to excessive memory usage. Tests with a slightly reduced set of input data showed that the proportion of rules in that case became approximately 52% with a single attribute, 41% with two attributes and only 7% with three attributes. This is further discussed in the section that follows.

6.2.3 Results for synthetic data

As the number of customers grow, through more facilities or more customers per facility, the time shows a comforting and almost perfect linear growth, while the number of rules grow slightly faster (figures 6.1(a) and 6.1(b)). The largest test cases had 250000 customers.

The time and number of rules grow rapidly as the number of categorical attributes increases. This can be motivated by the fact that if a customer group is a combination of up to \( s \) attributes (the default value for \( s \) was 2 in these tests) and the number of categorical attributes doubles, there will be roughly \( 2^s \) more ways to create rules. In practice, many of these combinations will be too small and pruned early. Yet, the growth rate is worrying if there are a large number of categorical attributes, and especially if it is desired to consider combinations of several of these attributes. A more detailed analysis shows that it is not the Apriori algorithm that
Figure 6.1. Elapsed time and number of rules generated by varying each of the parameters in table 6.5.
6.2. SCALABILITY

is slow – in fact the finding of large itemsets only accounts for about 5% of the total time regardless of the number of attributes – rather it is the search for rules and the numerous significance tests that take up most of the time.

In figure 6.1(d), all categorical attributes were given the same number of categories. The results seem to show that there are two peaks for which more rules are generated (and more time is required), one for around 10-15 categories and one for around 150-300. This relationship may seem puzzling at first, but there is a logical explanation. First of all, it can easily be realized that using fewer categories results in fewer combinations out of which to create rules. On the other hand, using too many categories means that there will not be as many groups of the minimum group size. Using 10-15 categories turns out to be the breakpoint where most categories are sufficiently large that when combined with another category, the combinations are still larger than the minimum group size. Similarly, 150-300 categories is the breakpoint where each individual category is larger than the minimum group size. For uniformly distributed data, these breakpoints can be found analytically as the quotient of the number of guests per facility and the minimum group size, and the square root of that quotient. For our experiment, that would be $\frac{500}{2} = 250$ and $\sqrt{500/2} \approx 15.8$. The number of rules tends to reach its local maxima slightly before the breakpoints since a larger portion of the potential rules are then found significant.

Figure 6.1(d) further shows that much more time is required for the second peak, even though fewer rules are generated. Contrary to the previous round of tests, this time the slowdown is caused by the Apriori algorithm. With many categories, Apriori generates a vast number of candidate sets, and ends up consuming as much as 80% of the execution time. Clearly, an alternative to Apriori, such as the FP-tree algorithm mentioned in subsection 3.2.1, would have been helpful in these cases. (However, none of the other experiments performed, with synthetic or real data, would have gained particularly much by that.)

The time and number of rules in figure 6.1(e) show a near linear growth in the number of quantitative attributes. This is entirely expected, since the discovery of rules is independent for each quantitative attribute. Note that the discovery of large itemsets is only run once regardless of the number of quantitative attributes; nonetheless, this stage only took about 5 seconds in these tests.

The run time grows almost quadratically as the number of quantitative levels increases. This can be explained by the fact that the significance tests take up an increasingly large portion of the total run time, and as mentioned in section 4.3.3, they execute in time proportional to the squared number of quantitative levels. If a fine level of discretization is required, a parametric alternative to the significance test is recommended, as discussed in 4.3.3.

Figure 6.1(g) shows something interesting: there is a sharp decrease in the number of rules as the significance level is lowered, yet the elapsed time is almost constant. The explanation is that the significance test is the last step performed before declaring that a relation is a rule, and that test needs to be done regardless of what the significance level is. Figure 6.1(h) shows, in a way, the opposite behav-
ior: the elapsed time decreases yet the number of rules is initially almost constant. This is the result of more relations being pruned due to not being different enough, consequently not having to go through a more costly significance test. The conclusion from these two experiments is that it is much more efficient to constrict a mining query by increasing the minimum mean difference than by decreasing the significance level. In fact, by increasing the minimum mean difference from 0.25 to 0.5, the elapsed time shrunk by 25%, yet only a single rule was lost.

When the minimum customer group size increased (figure 6.1(i)), the number of generated customer groups (which is not shown) decreased in a roughly inversely proportional manner. With a minimum group size of 2 customers, 40% of all groups had 2 customers and another 28% had 3 or 4 customers. Only 8% of the groups had 20 or more customers. The time and number of rules did not shrink as fast as the number of groups, suggesting that many of the groups were pruned at an early stage.

As more complicated rules are allowed, i.e. rules consisting of a combination of more categorical attributes, the mining time grows alarmingly, as seen in figure 6.1(j). It is however interesting to see that when allowing three attributes, only 274 extra rules were found (compared to 6511 rules with a single attribute and 8694 rules with two attributes). The results with authentic data from section 6.2.2 showed a similar proportion, although a slightly larger part of those rules had three attributes. The reason for this is that with more attributes, the uncertainty of how attributes interact increases, and the restrictive model introduced in section 4.3.2 is more likely to reject the rule.

The default setting is to allow a rule to have two categorical attributes, plus the hotel attribute and the recent trend attribute, which are treated differently. In [4], Aumann and Lindell performed experiments with real data and a domain expert rating quantitative rules as being interesting or not. In the experiment they performed, rules consisting of a single categorical attribute were found interesting approximately 3.5 times more frequently than rules with more attributes. Their conclusion was that compound rules are more difficult for a user to understand, and that most interesting rules are of simple nature. Due to the similarity of their problem and ours, it is our belief that their conclusions extend to our case, and we do not feel particularly limited to restricting a search to only include two simultaneous categorical attributes.

Not shown in the figures is the impact of mining recent trend rules. To test this, each response was given a random date ranged from the present day to one year back. The period used to look for recent trends was the last 90 days, and every single day was tested. 1863 trend rules were found in addition to the 15205 rules found with the default parameter settings. The time to mine all rules including the trend rules increased from 107 seconds to 193 seconds.
6.3 Learning the user’s interests

6.3.1 A user model

To evaluate the ability of an interestingness model to adapt to a user’s preferences, a model is used to simulate a user’s interests and feedback. This model, referred to as a user model, defines a target interestingness rating for each rule, and defines the feedback behavior of the user; which rules he chooses to rate and how he rates them. Once a few rules have been rated, the interestingness model is asked to predict the rating for the entire set of rules, and the predicted values can be compared to the known target values in order to construct a measure of the accuracy of the model.

The user model selected for this experiment defines the target rating function \( g(R) \) for a rule \( R \) as a power transformation of the number of rules \( n(R) \) in the rule database with the same rule components (categorical and quantitative attributes) as \( R \). \( R \) is included in the count, so \( n(R) \geq 1 \).

\[
g(R) = \frac{n(R)^\lambda - 1}{\max(n(R))^\lambda - 1}
\]

where \( \lambda \) is chosen to make the data more uniformly distributed (the untransformed distribution will typically be skewed to the left since many rules occur only once or a few times). Observe that \( 0 \leq g(R) \leq 1 \) (given that \( \lambda \neq 0 \)). The point of the target function is not to try to predict which rules are interesting to an actual user, but simply to assign an interestingness rating to each rule according to some system and see if the interestingness model can learn that system. The choice of a system is important for the results; for example, were we to assign a random rating to each rule, any model would have a very difficult time learning anything, and the results from such an experiment would not be very useful. The target function used here on the other hand tends to assign similar ratings to rules that are similar in nature.

The behavior of the user is defined as repeatedly selecting a random rule for which the user considers the currently predicted rating incorrect, i.e. having a predicted rating more than \( h \) units off from what the user would assign to that rule. The selected rule is then rated at the exact value given by the target function. It should be noted that an actual user is not expected to rate rules at such a precision, but this is the approach that will be used to be able to do a more precise comparison of the performance of different interestingness models.

6.3.2 The experiment

The same hotel data as in section 6.1 were used for this experiment. A total of 105810 rules were found for the settings used and \( \lambda = 0.1 \) was used to transform their counts into a roughly uniformly distributed target rating. The value \( h \) in the last paragraph was set to 0.1. Three interestingness models were compared: a Naive Bayes classifier, a multi-layer perceptron neural network and the memory based reasoning model described in chapter 5. For both the Naive Bayes classifier
and the neural network, the independent variables consisted of a binary list of all the quantitative and categorical attributes, with the value 1 meaning that the attribute was present in a rule and 0 (or \(-1\) for the neural network) that the attribute was not present. Two classes were used with the Naive Bayes classifier: interesting (I) and uninteresting (U). The probability that an attribute \(a_i\) belongs to the classes I and U was defined as the mean value of the ratings \(t_j\) that have been assigned to the rules \(R_j\) that do and do not contain the attribute \(a_i\), respectively:

\[
p(a_i|I) = \frac{\sum \{t_j \mid a_i \in R_j\}}{\text{count of } R_j}
\]

\[
p(a_i|U) = \frac{\sum \{t_j \mid a_i \notin R_j\}}{\text{count of } R_j}
\]

The predicted rating \(p(R)\) for a rule \(R\) can be seen as a weighted average of \(p(I|R)\) and \(p(U|R)\) with weights 1 (interesting) and 0 (uninteresting) respectively:

\[
p(R) = \frac{p(I|R)}{p(I|R) + p(U|R)}
\]  

(6.1)

Here, \(p(I|R)\) and \(p(U|R)\) are obtained using Bayes' theorem.

The neural network used the back-propagation learning technique with four hidden layers. The accuracy of the models was measured as the percentage of all rules for which the predicted rating differed at most \(d\) units from the target rating, with \(d = 0.1\) and \(d = 0.2\) used throughout all experiments.

For each model, 100 ratings were simulated, and after each rated rule, the accuracy of the model was computed. This procedure was repeated 20 times and the average accuracy for each model and number of rules is shown in figure 6.2.

![Figure 6.2](image)

\(\text{Figure 6.2. Accuracy for each model and for each number } k \text{ of rated rules.}\)
\(\text{The graphs show the average of 20 runs.}\)

### 6.3.3 Results

The memory based reasoning model suggested in this thesis was superior to both other methods for the experiment conducted. After 20 simulated user ratings, it
rated 52% of all rules within 0.1 units of the target rating and 81% within 0.2 units. After 100 ratings, those figures were 69% and 90%. The corresponding measures for the neural network were 33% and 54% after 20 rated rules and 55% and 80% after 100 rules. The Naive Bayes classifier initially performed better than the neural network, at 35% and 63% after 20 rules, but its accuracy did not improve much as new rules were rated and ended at 42% and 72%.

Another important aspect of an interestingness model is the stability of its predictions, and the MBR model was superior in this aspect too, as can be seen in the smoothness of the curves in figure 6.2. The neural network in particular was very instable, and its accuracy could change greatly as new rules were rated. One explanation for this is that its weight layers are given random start values, and thus it comes up with different solutions every time it is run.

The Naive Bayes classifier is almost on par with the MBR model as long as around 8 or fewer rules have been rated, but after that its pace of improvement slows markedly. This could be the effect of the assumption made in Naive Bayes that all attributes are independent of each other. Another possible explanation is that the task of predicting a numeric output value is not optimally suited to Naive Bayes, since even though the maximum probability would be assigned to the correct class (i.e., interesting or uninteresting), the probability measures need not be very accurate, which would make formula 6.1 less accurate [11]. One idea could thus be to skip the numeric interestingness measure and only use classes, e.g. \{A = Completely uninteresting, B = Fairly uninteresting, C = Neutral, D = Interesting, E = Highly interesting\}. A disadvantage of such an approach is that the classifier would not be aware of the underlying order between these classes; A would be considered as far away from B as from E. This idea has not been developed further here.

6.4 Feedback model

6.4.1 The experiment

In this experiment, the rule clustering idea described in 5.2.2 for quickly learning a user’s interests was tested. The test environment was the same as in section 6.3, except that the behavior of the user in the user model was modified to simulate that the user is asked by the system to rate a given list of rules. Two clustering algorithms were implemented and tested, the k-center algorithm used by Xin et al. in [35] and the k-means algorithm.

The k-center algorithm aims at minimizing the maximum distance from a rule to its nearest cluster center and was implemented by first selecting a random rule as the center of the first cluster, then repeatedly picking the rule having the maximum distance to the nearest already picked rule as the next cluster center. As proved in [12], this greedy algorithm provides a solution that is within a factor of 2 of the optimal solution.
The $k$-means algorithm [22] aims at minimizing an objective function given by

$$\sum_{j=1}^{k} \sum_{R_i \in c_j} D(R_i, R_{c_j})^2$$

where $D$ is a distance function, $c_j$ denotes the cluster $j$ and $R_{c_j}$ the rule representing the center of cluster $j$. Note that rules are used as center points here since the distance measure is based on rules. The $k$ cluster centers are initially placed randomly after which each rule is assigned to its nearest center. Once the $k$ clusters have been created, its center points are moved to the locations that minimize the local (intra-cluster) objective function. The cluster assignment and center relocation is repeated until the centers no longer move and a local minimum has been found.

For both algorithms, the user model was finally asked to rate the $k$ rules constituting the cluster centers. The algorithms were run 20 times, and in each run, $k$ assumed every integer value from 0 to 100. The random number generator was seeded with a unique value at each run, but the same seed was used for each value of $k$ so that the $k-1$ clusters from the test before would be initiated the same way (although for $k$-means clustering, the cluster centers would of course move during the center relocation phase). The point of this was to try to reduce the effect of randomness so that the effect of changing $k$ could be more easily observed. The results from the two clustering based approaches were compared to the random approach used in section 6.3 and are shown in figure 6.3.

![Figure 6.3](image)

Figure 6.3. Accuracy for the different feedback models and for each number $k$ of clusters/rated rules. The graphs show the average of 20 runs.

### 6.4.2 Results

Somewhat surprisingly, both clustering algorithms performed worse than rating rules at random. Let us first point out that rules can be thought of as existing in a feature space of high dimensionality, for which clustering is not optimally suited,
6.4. FEEDBACK MODEL

c.f. section 3.1. We believe that the bad performance of the $k$-center algorithm can be explained by the fact that the rule list, due to the high dimensionality, tends to contain many outliers: one or a few rules that are considered distant from all other rules. The $k$-center algorithm, attempting to minimize the maximum distance from a rule to its nearest cluster center, will thus typically assign clusters to outliers, which is not very helpful for the global rating accuracy.

The $k$-means algorithm performs almost as well as the random approach in estimating rule interestingness within 0.2 units, but is notably worse at estimating the interestingness more closely than so. One reason for this could be that in the random method, the user was assumed to browse around and only choose to rate rules for which he considered the current rating more than 0.1 units off; a behavior that obviously improves the accuracy in the case $d = 0.1$. This would not work for clustering-based algorithms, since it is the algorithm that chooses which rules should be rated, not the user. It is nevertheless possible that, for another user model, the results would have been different. An actual user is not likely to be rating rules completely at random, so a test with real users is strongly advised in order to evaluate which method works best in practice.

The odd spike seen in figure 6.3(b) for $k = 2$ is due to the fact that, for the given data, the two cluster centers usually ended up in the same locations regardless of their start locations, and rating the rules at these centers happened to result in an overall rating accuracy of 57% ($d = 0.2$).
Chapter 7

Conclusions

This thesis presented an efficient method for mining quantitative association rules for finitely discrete quantitative data. Our main contribution was a new view on what constitutes an unexpected relation, by taking into consideration the effects and interactions of categorical attributes. We believe this view to be more accurate than previous models. While the new model is guaranteed to reduce the amount of type I errors from its predecessors, there is also an increase in type II errors, and the lack of user tests currently prevents us from claiming the superiority of our model compared to others.

To ensure the statistical significance of the results, a non-parametric exact test was developed that does not require data to be normally distributed, works with sample sizes as low as a single sample and has a run time proportional to the sample size.

Experiments with survey data indicated that our data mining method scales well with the number of customers, quantitative attributes and categories, but less well with the number of categorical attributes. The latter is in line with what Brin et al. experienced in [7] when mining census data, and just like them, we do not have a good solution apart from advising an analyst to remove any unnecessary categorical attributes before mining.

In addition to mining rules, we showed how memory-based reasoning could be used to learn and to predict the users’ interests in rules. The method we suggested was compared to two other popular prediction methods, a Naive Bayes classifier and a neural network, and appears to be superior to both, both in terms of learning quickly and being able to reach a high prediction accuracy. All tests were however carried out in a simulated environment, and a natural conclusion is that user tests should be performed to confirm that the results hold in practice.

7.1 Generalization of work

Most of our work can be generalized to continuous quantitative data. However, the exact statistical test requires discrete data and experiments proved that it does not
scale very well with the number of discretization levels. If discretization is not an option or performs too slowly, it was suggested to use either a parametric test or a Monte Carlo method, depending on the sample size. Of course, if the quantitative data is normally distributed, a parametric test can replace the exact test entirely.

We showed how a quantitative attribute could be used as a rule antecedent; in our case we created an attribute to represent events that had happened within $n$ days, and explained how to efficiently find the integer value of $n$ that maximized a given measure. This procedure can be generalized to search for interesting intervals in any quantitative attribute, although the use of several such attributes simultaneously has not been experimented with here.

7.2 Fulfillment of purpose

We consider the purpose of this thesis mostly fulfilled. We believe that the method developed is able to detect all of the type of relations that were set out to be found. We also consider it to be largely autonomous and applicable to a wide variety of businesses. While a few steps of manual preparations for each newly set up business were advised to improve the results, we do believe that process to be less extensive than other tasks associated with setting up a new business. The manual preparations need only be performed once, after which the mining system can be set to run on a frequent basis, e.g. each night, to update the rule database. Whether the resulting rules can be presented directly to decision makers is more debatable. While much work was done to aid the user in locating interesting rules, the system is still of an interactive character, which according to e.g. [10] and [16] is an inherent property of knowledge discovery.

We initially criticized Chen’s work in [9] for leaving most of the work to the user, and although we too ended up on the same path regarding the use of visual aids such as box plots, our visual aids are merely intended for understanding and presenting more information about a rule that has already been found, and not for guiding the user to find rules in the first place.

In the experiments reported on in this thesis, a quarter of a million surveys were mined in around 15 minutes on the laptop test machine, and the scalability analysis shows that the method is clearly fit to tackle a million or more surveys.

7.3 Future work

There are several directions in which our work can be improved and many ideas that should be experimented with to see if they are fruitful. Probably the most important part that remains is to confirm the results claimed in this thesis by means of user tests.

The manual partitioning of quantitative independent variables into discrete categories may be considered onerous, and algorithmical solutions for this, such as the one described in [33], can be looked into.
7.3. FUTURE WORK

An interesting topic for further research is whether rule rating or rule ranking is the best way for a user to leave feedback. Both approaches have their pros and cons, but rule rating was ultimately chosen in this project since it worked better with our interestingness model. A related topic is to investigate to what extent rule ratings or rankings can be used to improve the accuracy of a rule distance measure.

Finally, we limited ourselves in this thesis to eschew text mining, but it is clear that free text data, if available, is a very valuable asset. Future work could thus be directed towards determining whether free text has a role to play in survey mining.
Bibliography


