University of Helsinki
Department of Computer Science
Series of Publications C, No. C-1998-42

Attribute Similarity and Event Sequence Similarity in Data Mining

Pirjo Ronkainen

Helsinki, October 1998

Report C-1998-42

University of Helsinki
Department of Computer Science
P. O. Box 26 (Teollisuuskatu 23)
FIN-00014 University of Helsinki, FINLAND

Attribute Similarity and Event Sequence Similarity in Data Mining

Pirjo Ronkainen

University of Helsinki, Department of Computer Science
Licentiate Thesis, Series of Publications C, Report C-1998-42

Helsinki, October 1998, 98 pages

Abstract

In data mining and knowledge discovery, similarity between objects is one of the central concepts. A measure of similarity can be user-defined, but an important problem is defining similarity on the basis of data. In this thesis we consider two kinds of similarity notions: similarity between binary valued attributes and between event sequences.

Traditional approaches for defining similarity between two attributes typically consider only the values of those two attributes, not the values of any other attributes in the relation. Such similarity measures are often useful, but unfortunately, they cannot reflect certain kinds of similarity. Therefore, we introduce a new attribute similarity measure that takes into account the values of the other attributes. The behavior of the different measures of attribute similarity is demonstrated by giving empirical results on two real-life data sets.

We also present a simple model for defining similarity between event sequences. The model is based on the idea that a similarity notion should somehow reflect how much work is needed in transforming an event sequence to another. We formalize this notion as edit distance between sequences. We show how the resulting measure of distance can be efficiently computed using a form of dynamic programming, and we also give some experimental results on two real-life data sets.

As one possibility of using the similarity notions discussed, we present how attributes and event sequences can be clustered to hierarchies. We describe three standard agglomerative hierarchical clustering methods, and give a set of clustering measures needed in finding the best clustering in the hierarchy of clusterings. The results of our experiments show that with these methods we can produce natural clusterings of attributes and event sequences.

Key Words:

Similarity, Distance, Clustering, Data mining, Knowledge Discovery

CR Classification:

H.3.3 Information Search and Retrieval

I.2.6 Artificial Intelligence: Learning

Contents

1 Introduction **1**

2 Similarity notions and their uses **4**

3 Similarity between attributes **8**

3.1 Attributes in relations 8
3.2 Internal measures of similarity 15
3.3 External measures of similarity 21
3.4 Algorithms for computing attribute similarity 27
3.5 Experiments 29
3.5.1 Data sets 29
3.5.2 Results and discussion 31

4 Similarity between event sequences **36**

4.1 Event sequences 36
4.2 Similarity measures 41
4.2.1 Event type sequences 45
4.2.2 Event sequences 47
4.3 Algorithm for event sequence similarity 53
4.4 Experiments 58
4.4.1 Data sets 58
4.4.2 Results and discussion 59

5 Clustering by similarity **64**

5.1 Hierarchical clustering 64
5.2 Clustering measures 66
5.2.1 Distance of clustering 66
5.2.2 Tightness of clustering 71
5.2.3 Quality of clustering 72
5.3 Algorithm for hierarchical clustering 73
5.4 Experiments 76
5.4.1 Clustering of attributes 76
5.4.2 Clustering of event sequences 83

6 Conclusions **89**

References **91**

1 Introduction

The rapid development of computer technology in last decades has made it possible for data systems to collect huge amounts of data. For example, a telecommunication network produces daily large amounts of alarm data. Analyzing such large data sets is tedious and costly, and thus, we need efficient methods to be able to understand how the data was generated, and what sort of patterns or regularities there exists in the data. A research area in computer science that considers this kind of questions is called *data mining*, or *knowledge discovery in databases* (KDD); see, e.g., [PSF91, FPSSU96] for overviews of research in data mining.

In order to find patterns or regularities in the data, it is necessary that we are able to describe how far from each other two data objects are. This is the reason why *similarity* between objects is one of the central concepts in data mining and knowledge discovery. During the last few years, there has been considerable interest in defining intuitive and easily computable measures of similarity between objects in different application areas and in using abstract similarity notions in querying databases [AFS93, APWZ95, ALSS95, CPZ97, GK95, JMM95, Ket97, KA96, KJF97, LB97, RM97, SK97, WJ96].

A typical data set considered in data mining consists of a number of data objects with several attributes. An example of such a data set is market basket data where data objects present customers and attributes are different products sold in the supermarket. Similar data sets occur, for example, in information retrieval: there the objects are documents and the attributes (key)words occurring in the documents. Even if the attributes in a database usually have a large value domain, for simplicity, we will consider in this thesis only binary valued attributes.

When discussing similarity and databases, one often talks about similarity between the objects stored in the database. For example, in market basket data, this would mean that we are interested in finding similarities between the customers of the supermarket. In such a case, the notion of similarity could, for instance, be used in customer segmentation or prediction. There exists, however, another class of similarity notions, i.e., *similarity between (binary) attributes*. In the market basket database setting, we could, for example, define similarity notions between the products sold in the supermarket by looking at how the customers buy these products.

One of the problems we consider in this thesis is the problem of defining similarity between attributes in large data sets. A traditional approach for attribute similarity is to use an *internal* measure of similarity. An internal measure of similarity between two attributes is defined purely in the terms of the values of these two attributes. Such measures of similarity are useful in several applications but, unfortunately, they are not able to reflect certain types of similarity. That is why we propose using an *external* measure of similarity between attributes. In addition to the values of the two attributes compared, an external measure takes into account also the values of a set of other attributes in the database. We contend that external measures can in several cases give more accurate and useful results than internal measures can.

A similarity notion between attributes can be used in forming hierarchies or clusters of attributes. Such a hierarchy describes the structure of the data, and can be used to form different kinds of rules such as generalized association rules [SA95], or characteristic rules and discrimination rules [HCC92]. Often the hierarchy of attributes is supposed to be given by an expert on the domain. Unfortunately, the domain expertise needed to form the hierarchy is not always available, and, hence, we need a way of computing the similarities and forming the hierarchies. Another reason for the need of a computable similarity notion between attributes is that we want to derive the similarity hierarchy on the basis of the actual data, not by using the apriori knowledge.

Another important form of data considered in data mining is sequential data. This kind of data occurs in many application domains, such as biostatistics, medicine, telecommunication, user interface studies, and WWW page request monitoring. Abstractly, such data can be viewed as a sequence of events where each event has an associated time of occurrence. During the last few years, interest to develop methods for knowledge discovery from sequences of events has increased; see, e.g., [AS95, HKM$^+$96, JCH95, Lai93, MTV95, MTV97, MKL95, OC96].

Analyzing sequences of events gives us important knowledge about the behavior and actions of a system or a user. Such knowledge can, for example, be used in locating problems and possibly predicting severe faults in a telecommunication network. In this thesis we consider the problem of defining *similarity between event sequences*. A lot of work has been done in the area of similarity between numerical sequences, but as far as we know, we are the first ones to consider similarity between event sequences. Our approach is based on the intuitive idea that similarity between event sequences should somehow reflect the amount of work that is needed to transform one event sequence to another. We formalize this notion as edit distance between sequences, and show that the resulting definition of similarity have several appealing features.

A similarity notion between event sequences can be used to build an index of a set of sequences. Such an index can then be used for finding efficiently all sequences similar to the pattern sequence given as a query to the database. On the other hand, we could be interested in predicting an occurrence of an event of a particular type in a sequence. For that we would have to find typical situations preceding an occurrence of an event of this type. These situations can be found, for example, by grouping all the sequences preceding the occurrences of such an event based on the similarities between the sequences.

The main aim of this thesis is to discuss similarity notions for data mining, especially in two particular cases of binary valued attributes and of event sequences. In addition to that, as one possible way of using the similarity notions we describe how both attributes and event sequences can be clustered by similarity to form hierarchies. In clustering we use three standard agglomerative hierarchical clustering methods. The best clustering among the hierachy of clusterings is chosen by using a set of intuitive clustering measures. We also give some experimental results on clustering of attributes and event sequences with the three clustering methods by using different similarity notions.

The rest of this thesis is organized as follows. First, in Chapter 2 we discuss some general properties and uses of similarity notions for data mining. Then Chapter 3 describes how we can define similarity notions for attributes in binary valued relations. In Chapter 4 we represent the main characteristics of event sequences and define similarity between event sequences as edit distance. After that Chapter 5 presents three hierarchical clustering methods and describes how the similarity notions defined in the earlier chapters can be used to form clusterings of similar attributes and event sequences. Finally, in Chapter 6, we make some concluding remarks and discuss future work.

2 Similarity notions and their uses

We start by discussing the meaning and uses of similarity notions. In this chapter we also describe some general properties that we expect a similarity notion to have.

Similarity is an important concept in many research areas. For example, in biology, computer science, linguistics, logic, mathematics, philosophy and statistics, a lot of work has been done on similarity. The main goal of data mining is to analyze data sets and find patterns and regularities that contain important knowledge about the data. In searching such regularities, it is usually not enough to consider only equality or inequality of data objects. Instead, we need to consider how similar two objects are, i.e., we have to be able to quantify how far from each other two objects are. This is the reason why similarity between objects is one of the central concepts in data mining and knowledge discovery.

A notion of similarity between objects is needed in virtually any database and knowledge discovery application. In the following there are some typical examples of such applications.

- Market basket data contains a lot of valuable information about customer behavior in terms of the purchased products. For example, information about products with similar selling patterns can be useful in planning marketing campaigns and promotions, in product pricing, or just in placement of products in the supermarket.

- In information retrieval a user typically wants to find all documents that are semantically similar, i.e., documents that are described by similar keywords. Therefore, in efficient retrieval we need both a notion for similarity between documents and a notion for similarity between keywords.

- In molecular biology one of the basic problems is a comparison of sequences where the idea is to find which parts of the sequences are alike. For example, the user can be interested in finding out how similar two DNA sequences of the same length are, or are there any subsequences in a long DNA sequence that are similar to a given short DNA sequence.

- "How similar are the sequences that precede occurrences of an alarm of type 1400?" or "are alarm sequences from Monday afternoon similar to sequences from Friday afternoon?" are two interesting questions that could be made in telecommunication alarm data. Answers to the same kind of questions could be needed in analysing any other sequential data, for example, a WWW page request log.

- From financial time series data a user may be interested in finding, for example, stocks that had last week a large price fluctuation, or identifying companies whose stock prices have similar pattern of growth. The same kind of queries one could pose in any other set of time-series.
- In image databases it can be interesting to retrieve all such images in the database that area similar to the query image, for example, with respect to certain colors or shapes in the images.

The examples above show clearly how important and essential the notion of similarity is for data mining. Searching for similar objects can help, for example, in predictions, hypothesis testing, and rule discovery rules [WJ96]. Moreover, a notion of similarity is needed and used in grouping and clustering of objects.

The meaning of similarity depends, however, largely on the type of the data. The objects considered in data mining are often complex, and they are described by different kind and number of features. It is, for example, clear that similarity between binary attributes is determined differently from similarity between images, or sounds. Neither can we define similarity between biosequences exactly the same way as similarity between time series. On the other hand, in one set of data we can have several kinds of similarity notions. Consider, for example, market basket data. In this data set, it would not be natural to define similarity between the customers in the same way as similarity between the products sold in the supermarket. Also in telecommunication data similarity between event types should not be determined with the same similarity notion as similarity between sequences of events.

The meaning of similarity may also vary depending on what kind of similarity we are looking for. Different similarity measures can reflect different facets of the data, and therefore, two objects can be determined to be very similar by one measure and very different by another measure. This means that we have to carefully choose one particular measure and hope that it gives proper results, or we have to try several measures on the data and then, by comparing the results given by these measures, choose the one that suits best our purposes.

Despite the fact that there is no single definition for similarity, we can still try to describe some properties that every notion of similarity should have. In the following we use an approach where similarity between objects is defined in terms of a complementary notion of *distance*.

Ideally, a measure d for distance between two objects should be a *metric*, i.e., satisfy the following conditions. For all objects γ_i, γ_j and γ_k we should have

1. $d(\gamma_i, \gamma_j) \geq 0,$
2. $d(\gamma_i, \gamma_j) = 0$ if and only if $\gamma_i = \gamma_j$
3. $d(\gamma_i, \gamma_j) = d(\gamma_j, \gamma_i)$
4. $d(\gamma_i, \gamma_k) \leq d(\gamma_i, \gamma_j) + d(\gamma_j, \gamma_k)$

The first condition above says that a distance measure d should always have a non-negative value, which is a very natural requirement. The same holds good the third requirement which states that a distance measure d should be *symmetric*.

The second requirement states that if the value of the distance between two objects is zero, then the objects compared should be identical. This requirement is quite natural but, unfortunately, it may in some cases be too restrictive. It can, for example, happen that the measurements used in computing the distance between two objects give zero as the value of the distance, even if the objects are distinct. These objects can be, however, considered to be identical from the application's point of view. Hence, it causes no problems if such a measure is used as a distance measure. Such a measure is called a *pseudometric* [Nii87] if it satisfies the condition

$$2'. \quad d(\gamma_i, \gamma_i) = 0$$

which is a weaker form of the second requirement and states that the distance of an object to itself should always be zero. Of course, a distance measure that is a pseudometric must satisfy the requirements 1, 3 and 4.

The fourth requirement above states that a distance measure should satisfy the *triangle inequality*. The need for this property may not be immediately obvious. Consider, however, the problem of searching for objects similar to an object γ_i from a large set of objects. Assume then that we know that the object γ_i is close to an object γ_j and the object γ_j is far from an object γ_k. Now the triangle inequality tells us that also the object γ_i must be far from the object γ_k, and we do not need to actually compute the distance between the objects γ_i and γ_k. This is a crucial property, if we want to access large sets of objects efficiently.

On the other hand, without the requirement that a distance measure should satisfy the triangle inequality, we can have a case where $d(\gamma_i, \gamma_j)$ and $d(\gamma_j, \gamma_k)$ are both small, but still the distance $d(\gamma_i, \gamma_k)$ could be large. Such a situation is, of course, undesirable. In order to obtain the property that an object γ_i is close to an object γ_k when we know that the object γ_i is close to an object γ_j and the object γ_j is close to an object γ_k, the distance measure may not always need to satisfy the triangle inequality. Instead, it can be sufficient for the distance measure to satisfy a *relaxed triangle inequality* [FS96], i.e., to satisfy the condition

$$4'. \quad d(\gamma_i, \gamma_k) \leq \alpha \cdot (d(\gamma_i, \gamma_j) + d(\gamma_j, \gamma_k))$$

where α is a constant that is not too large. Because it is not quite sure how useful such a property is in practice, we prefer distance measures that satisfy the exact triangle inequality.

In addition to being a metric or a pseudometric, a distance measure d should be in some sense natural and it should describe the facets of data that are thought to be interesting. Moreover, the measure should be easy and efficient to compute. If the size of the object set considered is reasonable, a quadratic algorithm in the number of objects can still be more or less acceptable. After all, the number of pairwise distances between the objects is quadratic in the number of objects. However, a cubic algorithm may already be too slow.

Because the distance between objects is a complementary notion of similarity between objects, a distance measure should also capture properly the notion of similarity. This means that if two objects are similar, then the distance between them should be small, and vice versa. This requirement is difficult to formalize, and because similarity notions are so dependent on the type of the data and the application domain, it is not possible to write down any set of requirements that would apply to all cases.

As stated earlier, in some cases we may have to try several distance measures on the data. In order to find the measure, that suits our purposes, we need to compare the results given by the different measures. In such cases, the actual numerical values of the measures are not important, only the relative order of the distance values is. Two distance measures can then be said to behave similarly if they keep the same relative order of the distance values. That is d and d' agree with each other in the sense that

$$d(\gamma_i, \gamma_k) < d(\gamma_j, \gamma_k) \text{ if and only if } d'(\gamma_i, \gamma_k) < d'(\gamma_j, \gamma_k)$$

for all γ_i, γ_j and γ_k. If the condition above does not hold good, the measures do not behave in the same way and they give a different view of the data.

In the following chapters we consider similarity between objects in two particular cases. First, in Chapter 3 we give some measures for defining similarity between binary valued attributes. Then we represent in Chapter 4 how similarity between event sequences could be determined. After that, in Chapter 5 we describe how these similarity notions can be used both in clustering of attributes and in clustering of event sequences.

3 Similarity between attributes

In this chapter we study how similarity between binary valued attributes in a relation could be defined. We consider two basic approaches for attribute similarity. An *internal measure* of similarity between two attributes is purely determined based on the values of those two attributes, not on any other attributes in the relation. An *external measure*, on the contrary, takes into account also the values of some or all the other attributes in the relation.

In Section 3.1 we define the basic concepts of attributes and relations used in this thesis. Section 3.2 presents internal and Section 3.3 external measures for similarity. Algorithms for computing attribute similarity measures defined are given in Section 3.4. Section 3.5 represents the results of our experiments with different data sets and different measures. Part of the material in this chapter has been published in [DMR97, DMR98].

3.1 Attributes in relations

A well-known and widely used way of describing the structure of a database is the *relational model* [AHV95, EN89, MR92, Vos91, Ull88]. In this model the data is represented as relations, i.e., tables where each row describes an object in the application area considered. In this chapter we consider the following data model resembling the relational model.

Definition 3.1 A schema $R = \{A_1, A_2, \ldots, A_m\}$ is a set of *binary attributes*, i.e., attributes with a domain $\{0, 1\}$. A *relation* r over R is a set of *m-tuples* $(a_1, a_2, \ldots, a_m)$ called *rows*. Given a row $\mathtt{t}$, the value of an attribute A_i is denoted by $\mathtt{t}[A_i]$. If there is no risk for confusion, we use a notation A_i for $\mathtt{t}[A_i] = 1$ and a notation $\overline{A_i}$ for $\mathtt{t}[A_i] = 0$. The number of attributes in the relation r is denoted by $|R| = m$, and the number of rows by $|r| = n$. Figure 3.1 presents a generic example of such a relation. □

This kind of data can be found in many areas. In this chapter we use in examples and experiments a *market basket data*, a collection of *keywords of newswire articles* and a *course enrollment data*.

Example 3.1 In a market basket market data attributes represent different products such as *beer, (potato) chips, milk, mustard,* and so on. Each row in the relation represents shopping baskets of customers in the supermarket. If a customer bought just beer and chips, the row describing his shopping basket has values $\mathtt{t}[\text{beer}] = 1$ and $\mathtt{t}[\text{chips}] = 1$, and value 0 for all the other attributes. A small example of market basket data is presented in Figure 3.2. From this relation we can, for instance, see that customers 5 and 12 bought mustard,

Row ID	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
t_1	1	0	0	0	0	1	0	1	0	0
t_2	1	1	1	1	0	1	0	0	1	1
t_3	1	0	1	0	1	0	0	1	1	0
t_4	0	0	1	0	0	1	0	1	1	1
t_5	0	1	1	1	0	0	1	0	1	1
...										
t_{1000}	1	0	1	1	0	1	0	1	0	1

Figure 3.1: An example relation r over the binary attributes $\{A_1, \ldots, A_{10}\}$.

sausage, and milk, and that customer 1 just purchased chips. The size of the example market basket relation is 12.

Note that in our data model we consider only whether a product was purchased or not. In the relational data model also attributes like the quantity and the price of the products purchased would be taken into account. □

Example 3.2 As another example data set we use the so-called Reuters-21578 categorization collection of newswire articles [Lew97]. The data set was modified so that each row in the relation corresponds to a newswire article, and attributes of the relation are all the possible keywords describing the articles. The keywords are divided in five different categories: economic subjects, exchanges, organizations, people and places. For example, keywords associated with an article with a title "Bahia Cocoa Review" are *cocoa, El Salvador, USA,* and *Uruguay*, and with an article with a title "Six killed in South African gold mine accident" *gold* and *South Africa*. A total of 19716 articles out of 21578 have at least one associated keyword. In our examples and experiments, the size of the Reuters data set is, therefore, considered to be 19716 rows. □

We use letters from the beginning of the alphabet like $A, B, C, \ldots$ to denote attributes, and letters from the end of the alphabet like X, Y, and Z to denote attribute sets of the form $X = \{A, \overline{B}, C, \ldots\}$. The attribute set can also be written as a concatenation of its elements, i.e., $X = A\overline{B}C$. A set of all attributes is denoted by R, relations by letter r, and rows of relations by letter t.

In a relation there can be hundreds, or even thousands of attributes but typically only few of them have a value 1 in a row, i.e., the relation is very sparse. Therefore, it can be useful to view the relation so that each row is a set of those attributes that have value 1 in that row. The example market basket data is presented in this manner in Figure 3.3, and the Reuters-21578 data set in Figure 3.4.

Customer	chips	mustard	sausage	beer	milk	Pepsi	Coke
t_1	1	0	0	0	0	0	0
t_2	0	1	1	0	0	0	0
t_3	1	0	0	0	1	0	0
t_4	1	0	0	1	0	0	1
t_5	0	1	1	0	1	0	0
t_6	1	0	0	1	1	0	1
t_7	0	1	1	0	0	1	0
t_8	1	0	0	0	1	1	0
t_9	0	1	1	1	0	0	1
t_{10}	1	0	0	1	0	0	0
t_{11}	0	1	1	0	1	1	0
t_{12}	0	1	1	0	1	0	0

Figure 3.2: An example of a market basket data.

Typically, we are not interested in every row of the relation at same time, but just a fraction of the rows. This leads us to the definition of a subrelation.

Definition 3.2 Let R be a set of attributes, and r a relation over R. A boolean expression θ, which is constructed from atomic formulae of the form "$\mathsf{t}[A] = 1$" and "$\mathsf{t}[A] = 0$", is called a *selection condition* on the rows of a relation r. A *subrelation* of r that consists of the rows statisfying the selection condition θ is denoted as $r_\theta = \sigma_\theta(r)$. For example, a subrelation where the attribute $A \in R$ has value 1, i.e., the rows where $\mathsf{t}[A] = 1$, is denoted by r_A. Similarly, we denote by $r_{\overline{A}}$ the subrelation of r where the attribute A has value 0 and the size of this subrelation with $|r_{\overline{A}}|$. □

Example 3.3 A subrelation of beer buyers r_{beer} in the example market basket data in Figure 3.2 has a size 4, and consists of rows $\mathsf{t}_4, \mathsf{t}_6, \mathsf{t}_9$, and t_{10}. On the other hand, a subrelation of non-milk buyers $r_{\overline{\text{milk}}}$ consists of rows $\mathsf{t}_1, \mathsf{t}_2, \mathsf{t}_4, \mathsf{t}_7, \mathsf{t}_9$ and t_{10}, and its size, therefore, is 6. □

Example 3.4 A subrelation r_{USA} of the rows where keyword *USA* occurs in the Reuters-21578 data set has a size 12541, and a subrelation $r_{\text{El Salvador}}$ of rows with keyword *El Salvador* a size 11. On the other hand, a subrelation $r_{\overline{\text{Switzerland}}}$ consists of 19 502 rows (if also the rows without any keywords were considered, the size of $r_{\overline{\text{Switzerland}}}$ would be 21 364 rows). □

The size of a subrelation indicates the number of rows satisfying the given selection condition. Often we are not interested in the absolute number of rows

Customer	Purchases
t_1	{chips}
t_2	{mustard, sausage}
t_3	{chips, milk}
t_4	{chips, beer, Coke}
t_5	{mustard, sausage, milk}
t_6	{chips, beer, milk, Coke}
t_7	{mustard, sausage, Pepsi}
t_8	{chips, milk, Pepsi}
t_9	{mustard, sausage, beer, Coke}
t_{10}	{chips, beer}
t_{11}	{mustard, sausage, milk, Pepsi}
t_{12}	{mustard, sausage, milk}

Figure 3.3: The example market basket data in Figure 3.2 viewed in a form where each row consists of a row identifier and a set of the products purchased.

but rather would like to consider the relative number of the rows, their relative frequency.

Definition 3.3 Let R be a set of attributes, r a relation over R, θ a selection condition, and r_θ a subrelation satisfying the condition θ. The *frequency* of the subrelation r_θ, i.e.,

$$\frac{|r_\theta|}{|r|},$$

is denoted by $fr(\theta, r)$. If the relation r is clear from the context, we may write $fr(\theta)$. Additionally, we use the abbreviation $fr(A)$ for $fr(\mathsf{t}[A] = 1)$, and $fr(ABC)$ for $fr(\mathsf{t}[A] = 1 \wedge \mathsf{t}[B] = 1 \wedge \mathsf{t}[C] = 1)$. Similarly, $fr(\mathsf{t}[A] = 0)$ is denoted as $fr(\overline{A})$ and $fr(\mathsf{t}[A] = 0 \wedge \mathsf{t}[B] = 1 \wedge \mathsf{t}[C] = 0)$ as $fr(\overline{A}B\overline{C})$. We are usually interested only in the presence of attributes, i.e., the cases where $\mathsf{t}[A] = 1$, and therefore, we talk about the frequency $fr(X)$ of an attribute set X. □

Example 3.5 Let us consider the subrelations in Example 3.3. The frequency of beer buyers is $fr(\text{beer}) = 4/12 = 0.33$, and the frequency of non-milk buyers $fr(\overline{\text{milk}}) = 6/12 = 0.5$. □

Example 3.6 In the Reuters-21578 data set the most frequent keyword is *USA* with the frequency of 0.636. For the other keywords in Example 3.4 we have frequencies $fr(\text{El Salvador}) = 0.0006$, and $fr(\overline{\text{Switzerland}}) = 0.9891$. □

Article	Keywords
1	{cocoa, El Salvador, USA, Uruguay}
2	{USA}
3	{USA}
4	{USA, Brazil}
5	{grain, wheat, corn, barley, oat, sorghum, USA}
...	
21576	{gold, South Africa}
21577	{Switzerland}
21578	{USA, amex}

Figure 3.4: A part of Reuters-21578 collection viewed in a form of sets of keywords associated with the articles.

In Definition 3.3 we presented that the frequency can be defined for both the presence and the absence of the attribute i.e., the cases of $\mathsf{t}[A] = 1$ and $\mathsf{t}[A] = 0$, respectively. Usually, in literature and programs computing the frequencies only the presence of attributes is considered (see [SVA97] for an exception). This is, however no problem because the frequencies for attributes in absence can be computed from the frequencies of attributes in presence, i.e., $fr(\overline{A}) = 1 - fr(A)$. More about computing the frequencies can be read, for example, from [AIS93, AMS+96, Toi96].

The notion of frequency makes it possible for us to define association rules. An association rule describes how a set of attributes tends to occur in the same rows with another set of attributes.

Definition 3.4 An *association rule* in a relation r is an expression $X \Rightarrow Y$, where $X \subseteq R$ and $Y \subseteq R$. The frequency or *support* of the rule is $fr(X \cup Y, r)$, and the *confidence* of the rule is

$$conf(X \Rightarrow Y, r) = \frac{fr(X \cup Y, r)}{fr(X, r)}.$$

If the relation r is clear from the context, we write simply $fr(X \cup Y)$ and $conf(X \Rightarrow Y)$. □

The frequency of an association rule is the positive evidence for the rule in the relation r. The confidence of the rule is the conditional probability that a randomly chosen row from r that matches X matches also Y. Algorithms for computing association rules are described, for example, in [AIS93] and [Toi96]. Note that the right-hand side of an association rule was defined to be a set of attributes. In this thesis we consider, however, only association rules where the right-hand side of the rule is one attribute.

Example 3.7 Consider the example market basket relation in Figure 3.2. In that relation, for instance, the frequency of sausage buyers is $fr(\text{sausage}) = 0.5$, and the frequency of mustard buyers $fr(\text{mustard}) = 0.5$. The frequency of the rule “sausage $\Rightarrow$ mustard” is also 0.5 and the confidence of the rule is $0.5/0.5 = 1.0$. This means that every customer that bought mustard, also bought sausage, and vice versa.

The frequency of chips buyers in the same relation is $fr(\text{chips}) = 0.5$ and the frequency of beer buyers $fr(\text{beer}) = 0.33$. The frequency of the rule “beer $\Rightarrow$ chips” is 0.25 and the confidence of the rule is $0.25/0.33 = 0.75$. This means that 25 % of the customers bought both beer and chips, and 75 % of those whose bought beer bought also chips. On the other hand, the frequency of customers buying chips but not beer is $fr(\mathsf{t}[\text{beer}] = 0 \wedge \mathsf{t}[\text{chips}] = 1) = 0.25$, and the frequency of customers buying beer but not chips is $fr(\mathsf{t}[\text{beer}] = 1 \wedge \mathsf{t}[\text{chips}] = 0) = 0.083$. So the confidence of the rule “chips $\Rightarrow \overline{\text{beer}}$” is $0.25/0.50 = 0.50$ and the confidence of the rule “beer $\Rightarrow \overline{\text{chips}}$” is $0.083/0.33 = 0.25$. $\square$

Example 3.8 In the Reuters-21578 article collection the frequency of the keyword *grain* is 0.0319. The frequency of the rule “USA $\Rightarrow$ grain” is 0.0185 and the confidence of the rule is $0.0185/0.636 = 0.029$. The frequency of the rule “El Salvador $\Rightarrow$ grain“ is 0.0001 and the confidence of the rule is $0.0001/0.0006 = 0.167$. This means that 1.85 % of the articles have both the keywords USA and grain, and of the articles with the keyword USA only about 3 % have the keyword grain. The keywords El Salvador and grain occur very seldom in a same article, but as much as 16.7 % of the articles talking about El Salvador also mention the keyword grain.

The frequency of the rule “$\overline{\text{Switzerland}} \Rightarrow$ grain” is $fr(\text{grain}, \overline{\text{Switzerland}}) = fr(\text{grain}) - fr(\text{grain}, \text{Switzerland}) = 0.0319 - 0.00005 = 0.03185$ and the confidence of the rule is $0.03185/0.9891 = 0.0322$. On the other hand, the confidence of the rule “grain $\Rightarrow \overline{\text{Switzerland}}$” is $0.03185/0.0319 = 0.9984$. So if we know that the keyword grain is associated with the article, it is very unlikely that the keyword Switzerland is also associated with the article. But if we know that Switzerland is not a keyword of the article considered, we have about 3 % possibility that the keyword grain occurs among the keywords. When looking the actual articles we can see that there is only one article to which they both are associated. $\square$

Because there typically are tens or hundreds, even thousands of attributes, computing all the pairwise attribute similarities at the same time is tedious. On the other, in many cases we are not even interesting in finding similarities between all the attributes in a relation. This means that first we have to define which attributes in a relation interest us and then compute similarities just between these attributes.

Definition 3.5 A set of attributes $A_i \in R$ between which we want to compute similarity values is called *the set of interesting attributes*, and it is denoted by $\mathcal{AI}$. $\square$

The selection of the interesting attributes depends on the situation and application we are considering. A natural requirement is that the interesting attributes should be somehow intuitively similar. For example, we might think that *juice* and *washing powder* are not intuitively very similar products in the market basket data. Despite that, in some cases they might still be considered belonging to the same group of products, the group of food and household goods. When choosing the interesting attributes, one should also remember that if we consider just the attributes known to be associated with each other, some interesting and new associations between attributes may be lost.

Example 3.9 In market basket data a set of interesting attributes could consist of a set of beverages, dairy products, or fast food products. For example, products like *butter, cheese, milk* and *yogurt* could be the set of interesting dairy products, and *beer, milk, Coke* and *Pepsi* the set of interesting beverages. $\square$

Example 3.10 In the Reuters-21578 data a set of interesting places could be, for example, the set of keywords *Argentina, Canada, Spain, Switzerland, USA,* and *Uruguay.* On the other hand, keywords *Chirac, Deng Xiaoping, Gandhi, Kohl, Nakasone,* and *Reagan* could form the set of interesting people. $\square$

After the set of interesting attributes has been selected, we can compute the similarities between all pairs of attributes in the set. As stated in Chapter 2, we can define similarity between attributes also in terms of a complementary notion of distance between attributes. In such a case we have the following.

Definition 3.6 Given a set of attributes R and a class of all relations $\mathcal{R}$ over R, d is a *distance* function defined as $d : R \times R \times \mathcal{R} \to \mathbb{R}$. Given a relation r over an attribute set R, and attributes $A \in R$ and $B \in R$, the distance between these two attributes is denoted by $d(A, B; r)$. If there is no risk for confusion, we write just $d(A, B)$. $\square$

The exact choice of the similarity measure, obviously, depends often on the application and the type of similarity we are looking for. In the next two sections we consider two different approaches for computing the distance between attributes. In Section 3.2 we discuss internal distance measures and in Section 3.3 external distance measures.

3.2 Internal measures of similarity

An internal measure of similarity is a measure whose value for attributes A and B is only based on the values of the columns of these attributes. So these measures describe how two attributes A and B appear together, i.e., how they are associated with each other.

The statistics needed by any internal distance measure can be expressed by the familiar 2-by-2 contingency table given in Figure 3.5. The value n_{11} in the table describes the number of the rows in the relation that fulfil the condition "$\mathtt{t}[A] = 1 \wedge \mathtt{t}[B] = 1$", the value n_{10} the number of the rows that fulfil the condition "$\mathtt{t}[A] = 1 \wedge \mathtt{t}[B] = 0$", etc. This simplification is possible for two reasons. For the first, there are only 4 possible value combinations of the attributes A and B, and secondly, we are assuming that the order of the rows in the relation r does not make any difference. When the marginal proportions of the attributes are known, fixing of one cell value in the 2-by-2 contingency table fixes all the other cell values. This means that only one cell value can be assigned at will, and therefore, we say that any internal measure of similarity has 1 *degree of freedom*.

There are, of course, numerous ways of defining measures for the strength of association between attributes; see [GK79] for some possible measures. One of the possibilities is the χ^2 test statistic, which measures the deviation between the observed and expected values of the cells in the contingency table under the independence assumption. In the case of two binary attributes, the χ^2 test statistic is

$$\chi^2 = \sum_{i \in \{0,1\}} \sum_{j \in \{0,1\}} \frac{(n_{ij} - (n_{i\cdot} \cdot n_{\cdot j}/n))^2}{n_{i\cdot} \cdot n_{\cdot j}/n}$$

where n_{ij} represents the observed and $(n_{i\cdot} \cdot n_{\cdot j}/n)$ the expected number of rows with $\mathtt{t}[A] = i \wedge \mathtt{t}[B] = j$. With the attribute frequencies this measure can be expressed as

$$\chi^2 = n \cdot \sum_{i \in \{0,1\}} \sum_{j \in \{0,1\}} \frac{fr(ij)^2}{fr(i)\, fr(j)} - n$$

where the index i describes the values of the attribute A and the index j the values of the attribute B. As a measure of association between attributes we could also use any of the many modifications of the χ^2 test statistic, like Yule's, Pearson's or Tschuprow's coefficients of association [YK58, GK79].

The χ^2 test statistic determines whether two attributes A and B are independent or not. If the attributes are independent, the value of the measure is 0. When the value of χ^2 is higher than a cutoff value, the attributes are considered to be somehow dependent on each other. The cutoff value at the given significance level can be obtained from the common table of the significant points of χ^2 available in nearly every book of statistics.

	B	$\overline{B}$	$\sum$
A	n_{11}	n_{10}	$n_{1\cdot}$
$\overline{A}$	n_{01}	n_{00}	$n_{0\cdot}$
$\sum$	$n_{\cdot 1}$	$n_{\cdot 0}$	n

Figure 3.5: The 2-by-2 contingency table of the attributes A and B.

Example 3.11 Consider products *beer* and *milk* in the example market basket data. For these attribute we have the following contingency table (a) and the table of the expected values

(a)

	milk	$\overline{\text{milk}}$	$\sum$
beer	1	3	4
$\overline{\text{beer}}$	5	3	8
$\sum$	6	6	12

(b)

	milk	$\overline{\text{milk}}$	$\sum$
beer	2	2	4
$\overline{\text{beer}}$	4	4	8
$\sum$	6	6	12

With the values above, the χ^2 test statistic is

$$\chi^2 = \frac{(1-2)^2}{2} + \frac{(3-2)^2}{2} + \frac{(5-4)^2}{4} + \frac{(3-4)^2}{4} = 1.5.$$

At the 5 % significance level and with 1 degree of freedom, the cutoff value of χ^2 is 3.84. Because $1.5 < 3.84$, the products beer and milk are said to be independent at this significance level.

Consider then products *beer* and *Coke* in the same relation. For these products the contingency table (a) and the table of the expected values (b) are as follows

(a)

	Coke	$\overline{\text{Coke}}$	$\sum$
beer	3	1	4
$\overline{\text{beer}}$	0	8	8
$\sum$	3	9	12

(b)

	Coke	$\overline{\text{Coke}}$	$\sum$
beer	1	3	4
$\overline{\text{beer}}$	2	6	8
$\sum$	3	9	12

Now the value of the χ^2 test statistic is

$$\chi^2 = \frac{(3-1)^2}{1} + \frac{(1-3)^2}{3} + \frac{(0-2)^2}{2} + \frac{(8-6)^2}{6} = 8.$$

Because $8 > 3.84$, the products beer and Coke are dependent on each other at the 5 % significance level. □

Example 3.12 In the Reuters-21578 data the contingency table for keywords *USA* and *Switzerland* is

	Switzerland	$\overline{\text{Switzerland}}$	$\sum$
USA	35	12506	12541
$\overline{\text{USA}}$	179	6996	7175
$\sum$	214	19502	19716

The table of expected values for the number of occurrences of these keywords is

	Switzerland	$\overline{\text{Switzerland}}$	$\sum$
USA	136.12	12404.88	12541
$\overline{\text{USA}}$	77.88	7097.12	7175
$\sum$	214	19502	19716

The value of the χ^2 test statistic for these keywords is nearly 209. Thus, the observed significance level is very small, and the keywords can be said to be dependent on each other at any reasonable confidence interval.

On the other hand, for keywords *USA* and *El Salvador* the contingency table is

	El Salvador	$\overline{\text{El Salvador}}$	$\sum$
USA	7	12534	12541
$\overline{\text{USA}}$	4	7171	7175
$\sum$	11	19705	19716

which is the same as the table of the expected values. In this case the value of the χ^2 test statistic is 0, and the keywords are said to be independent. $\square$

The good thing with measuring significance of associations via the χ^2 test statistic is that the measure takes into account both the presence and the absence of attributes [BMS97, SBM98]. Unfortunately, as [GK79] puts it, "The fact that an excellent test of independence may be based on χ^2 does not at all mean that χ^2, or some simple function of it, is an appropriate measure of degree of association". One of the well-known problems with χ^2 is that using it is recommended only if all cells in the contingency table have expected values greater than 1, i.e., expected frequencies are large enough. Also at least 80 per cent of the cells in the contingency table have expected value greater than 5. In the example market basket data, for example, this causes difficulties. In addition, the total number of rows n considered should be reasonably large.

Because of the problems with the χ^2 measure, we consider here some other possibilities of defining an internal similarity measure. We start by defining a similarity measure that is based on the frequencies of the attributes.

Definition 3.7 Given the attributes A and $B \in R$, the *internal distance* $d_{I_{sd}}$ between them is defined as

$$\begin{aligned} d_{I_{sd}}(A,B) &= \frac{fr((\mathtt{t}[A]=1 \wedge \mathtt{t}[B]=0) \vee (\mathtt{t}[A]=0 \wedge \mathtt{t}[B]=1))}{fr(\mathtt{t}[A]=1 \vee \mathtt{t}[B]=1)} \\ &= \frac{fr(A)+fr(B)-2fr(AB)}{fr(A)+fr(B)-fr(AB)}. \end{aligned}$$

□

The similarity measure $d_{I_{sd}}$ focuses on the positive information of the presence of the attributes A and B. It describes the relative size of the symmetric difference of the rows with $\mathtt{t}[A] = 1$ and $\mathtt{t}[B] = 1$. The distance measure $d_{I_{sd}}$ is a complement of the well-known noninvariant coefficient for binary data, the *Jaccard's coefficient* [And73, KR90]:

$$\frac{fr(AB)}{fr(A)+fr(B)-fr(AB)}.$$

According to [MS68], the complement measure $d_{I_{sd}}$ can be shown to be a metric.

The values of the $d_{I_{sd}}$ measure vary between 0 and 1. The extremes of the value range of this measure can be reached as follows. If the attributes A and B are equally frequent and also the frequency of AB is the same, i.e., $fr(A) = fr(B) = fr(AB)$, the value of $d_{I_{sd}}$ is 0, and the attributes are said to be exactly similar. The attributes are considered as totally dissimilar, i.e., $d_{I_{sd}} = 1$, when $fr(A,B) = 0$.

Example 3.13 If we consider the products *beer* and *milk* in the example of market basket data, we notice that the frequency of beer buyers is $fr(\text{beer}) = 0.33$, the frequency of milk buyers $fr(\text{milk}) = 0.50$, and the frequency of customers buying both beer and milk is $fr(\text{beer}, \text{milk}) = 0.083$. Using these values we get that the distance $d_{I_{sd}}$ of beer and milk is $\frac{0.33+0.50-2\cdot 0.083}{0.33+0.50-0.083} = 0.89$. Hence, according to this measure beer buyers do not tend to buy milk, and vice versa. In this sense the beer buyers behave differently to milk buyers.

The frequency of Coke buyers in the example market basket data is 0.25 and the frequency of customers buying both beer and Coke is 0.25. The internal distance $d_{I_{sd}}$ for these products is $\frac{0.33+0.25-2\cdot 0.25}{0.33+0.25-0.25} = 0.25$. Thus, we can say that the buyers of Coke behave rather similarly to the customers buying beer. □

Example 3.14 In the Reuters-21578 data the frequency of the keyword *Switzerland* is 0.0109 and the frequency of the keyword *USA* is 0.636. The frequency of the attribute set $\{\text{Switzerland}, \text{USA}\}$ is 0.0018. With these values we get the distance $d_{I_{sd}}(\text{Switzerland}, \text{USA}) = 0.997$. The frequencies $fr(\text{El Salvador}, \text{USA}) = 0.0004$ and $fr(\text{El Salvador}, \text{Switzerland}) = 0.0$ in their turn give us distances $d_{I_{sd}}(\text{El Salvador}, \text{USA}) = 0.999$ and

$d_{I_{sd}}(\text{El Salvador}, \text{Switzerland}) = 1$. All the distances have quite high values which indicates that these keywords as such are different from each other and they do not appear in the same articles. $\square$

In data mining contexts, it would be natural to make use of association rules in defining similarity between attributes. One possibility is to define such a measure based on the confidences of the association rules $A \Rightarrow B$ and $B \Rightarrow A$.

Definition 3.8 Consider two attributes A and $B \in R$ and two association rules $A \Rightarrow B$ and $B \Rightarrow A$. The *internal distance* $d_{I_{conf}}$ between attributes A and B is defined as

$$d_{I_{conf}}(A, B) = (1 - \mathit{conf}(A \Rightarrow B)) + (1 - \mathit{conf}(B \Rightarrow A)). \quad \square$$

The internal distance measure $d_{I_{conf}}$ resembles the common *Manhattan distance* [KR90, Nii87], which is known to be a metric. The measure $d_{I_{conf}}$ is, however, only a pseudometric because its value can be zero even if the attributes A and B are not identical, i.e., $A \neq B$. This happens when the attributes A and B occur only in the same rows of the relation r.

The value range of $d_{I_{conf}}$ is $[0, 2]$ indicating that attributes that are similar have distance 0 and attributes that are totally dissimilar have distance 2. This means that in the former case the confidences of the association rules have value 1 which happens only when the attributes always occur in the same rows, i.e., $\mathit{fr}(A) = \mathit{fr}(B) = \mathit{fr}(AB)$. In the later case both the confidences are 0 which means that $\mathit{fr}(AB) = 0$ and the attributes A and B never have value 1 in a same row.

Example 3.15 Consider the buyers of *beer*, *milk* and *Coke* in the example market basket data. The internal distance $d_{I_{conf}}$ between beer and milk buyers is $d_{I_{conf}}(\text{beer}, \text{milk}) = (1 - 0.25) + (1 - 0.1667) = 1.58$. Thus, this measure indicates that beer and milk buyers behave differently. On the other hand, the internal distance $d_{I_{conf}}$ between beer and Coke buyers is $d_{I_{conf}}(\text{beer}, \text{Coke}) = (1 - 0.75) + (1 - 1) = 0.25$. Therefore, the buyers of these two beverages can be said to behave rather similarly. $\square$

Example 3.16 The distance $d_{I_{conf}}$ between keywords *Switzerland* and *USA* is 1.832, between keywords *El Salvador* and *USA* 1.399, and between keywords *El Salvador* and *Switzerland* 2. Therefore, the internal distance measure $d_{I_{conf}}$ indicates that these keywords are not behaving similarly and are only seldom associated with the same articles. $\square$

Internal measures represent the more traditional way of defining attribute similarity, and they are useful in many applications. Unfortunately, because they are based solely on the values of just the two attributes considered, they do not necessarily find certain types of similarity.

Example 3.17 Let *chips*, *milk*, *mustard* and *sausage* be four interesting products in the example market basket data. These products have some very interesting connections. The products chips and sausage, for example, are *substitutes* to each other because customers buying chips never buy sausages at the same time. On the other hand, customers buying mustard always buy sausages, and vice versa. Therefore, the products mustard and sausage are *complements* to each other. A third pair of products, milk and sausage, seem to be *independent* of each other, since milk buyers purchase sausage as often as non-milk buyers. Similarly, sausage buyers purchase milk as often as non-sausage buyers. The contingency tables of these three situations are given in Figure 3.6.

The three internal distance measures considered in this section give the following values for the pairs of products above.

Measure	chips and sausage	mustard and sausage	milk and sausage
χ^2	12	12	0
$d_{I_{sd}}$	1	0	2/3
$d_{I_{conf}}$	2	0	1

According the χ^2 test statistic the products milk and sausage are, indeed, statistically independent. In the two other pairs the products are dependent on each other: the products chips and sausage are completely negatively and the products mustard and sausage completely positively associated with each other. The measure $d_{I_{sd}}$, on the other hand, says that the product chips is totally dissimilar to the product sausage, the products milk and sausage are rather different from each other, and that the products mustard and sausage are exactly similar. The results given by the measure $d_{I_{conf}}$ are much the same as with the measure $d_{I_{sd}}$. □

The results of Example 3.17 can be generalized to every situation where the contingency tables are similar to Figure 3.6. Only the value of the χ^2 test statistic changes in the case of completely positively and negatively associated attributes: the value is always the number of rows in the relation r considered.

None of the internal measures is able to view the three types of situations above as reflecting that two attributes A and B are similar. Still, the similarity between the attributes A and B in each case can be high. The similarity between the attributes may be due to some other factors than just the information given by the values in the columns A and B. Therefore, we need to consider external measures for similarity. For them, also the values of other attributes than A and B have an influence on the similarity notion.

a)

	sausage	$\overline{\text{sausage}}$	$\sum$
chips	0	6	6
$\overline{\text{chips}}$	6	0	6
$\sum$	6	6	12

b)

	sausage	$\overline{\text{sausage}}$	$\sum$
mustard	6	0	6
$\overline{\text{mustard}}$	0	6	6
$\sum$	6	6	12

c)

	sausage	$\overline{\text{sausage}}$	$\sum$
milk	3	3	6
$\overline{\text{milk}}$	3	3	6
$\sum$	6	6	12

Figure 3.6: The 2-by-2 contingency tables for a) substitute, b) complement, and 3) independent products.

3.3 External measures of similarity

An external measure of similarity takes into account, of course, the values of attributes A and B, but also the values of other attributes, or a subset of other attributes. Using such measures we can find that two attributes A and B behave similarly even if they never occur in a same row of the relation r.

Example 3.18 In a market basket data, two products may be classified as similar if the behavior of the customers buying them is similar with respect to other products. For instance, two products, *Pepsi* and *Coke*, could be deemed similar if the customers buying them behave similarly with respect to products *mustard* and *sausage*. □

Example 3.19 Two keywords in the Reuters-21578 data set could be defined similar if they occur in a similar way with respect to a set of other keywords. Thus, keyword *El Salvador* and *USA* could be deemed similar if they are associated a similar way to keywords *coffee* and *grain*. □

The main idea in external measures is to define the similarity between attributes A and B by the similarity between subrelations r_A and r_B. An external measure should say that the attributes A and B are similar only if the differences between subrelations r_A and r_B can arise by chance. Similarity between these subrelations is defined by considering the marginal frequencies of a selected subset of other attributes in the relation.

Definition 3.9 A *probe set* $P \subseteq R$ is a collection of attributes of the relation r. We call the attributes D_i in $P = \{D_1, D_2, \ldots, D_k\}$ as *probe attributes.* Given the relation r and the probe set P, an *external measure of similarity* says that the attributes A and B are similar if the subrelations r_A and r_B are similar with respect to P. □

The probe set defines the viewpoint from which the similarity is judged. Thus, different selections of probe attributes produce different measures. The choice of the probe set is considered more later in this section.

We wanted originally to define similarity between two attributes of size $n \times 1$, and now we reduce this to similarity between two subrelations of sizes $n_A \times k$ and $n_B \times k$, where k is the number of attributes in P, $n_A = |r_A|$, and $n_B = |r_B|$. This may seem to be a step backwards, but fortunately, the problem can be diminished by using some very well-established notions, such as frequencies, in defining similarity between subrelations.

The subrelations r_A and r_B projected to the probe set P can be viewed as defining two multivariate distributions g_A and g_B on $\{0,1\}^P$. Then, given an element $x \in \{0,1\}^P$, the value $g_A(x)$ is the relative frequency of x in the relation r_A. One widely used distance notion between distributions is the *Kullbach-Leibler distance* [KL51, Kul59, Bas89]:

$$d(g_A \parallel g_B) = -\sum_x g_A(x) \cdot \log \frac{g_A(x)}{g_B(x)}$$

or the symmetrized version of it: $d(g_A \parallel g_B) + d(g_B \parallel g_A)$ when subrelations r_A and r_B are projected to P. This measure is also known as *relative entropy* or *cross entropy.* The problem with the Kullbach-Leibler distance is that the sum has $2^{|P|}$ elements, so direct computation of the measure is not feasible. Therefore, we look for simpler measures that would still somehow reflect the distance between g_A and g_B.

Basic measure

One way to remove the exponential dependency on $|P|$ is to look at only a single attribute $D \in P$ at a time. Thus, the similarity between attributes A and B can be defined as follows.

Definition 3.10 Given attributes A and B, and a probe attribute $D \in P$, a function for measuring how similar A and B are with respect to the probe attribute D is denoted by $E_{\mathsf{f}}(A, B, D)$. The *external distance* $d_{E_{\mathsf{f}},P}$ is defined as

$$d_{E_{\mathsf{f}},P}(A, B) = \sum_{D \in P} E_{\mathsf{f}}(A, B, D),$$

that is, as the sum of the values of E_{f} over all the probes $D \in P$. □

The $d_{E_{\mathsf{f}},P}$ measure, of course, is a simplification that looses power compared to the full relative entropy measure. Still, we suggest the measure $d_{E_{\mathsf{f}},P}$ as the external distance between attributes A and B. If the value of $d_{E_{\mathsf{f}},P}(A, B)$ is small, the attributes A and B are said to be similar with respect to the attributes in P. On the other hand, we know that attributes A and B are not behaving in the same way with respect to the attributes in P, if the value $d_{E_{\mathsf{f}},P}(A, B)$ is large.

Note that the function $E_{\mathsf{f}}(A, B, D)$ in Definition 3.10 was not fixed. This means that there are several different functions E_{f} for measuring similarity between subrelations r_A and r_B with respect to a probe attribute D. One possibility is to measure how different the frequency of D is in relations r_A and r_B. A simple test for this is to use χ^2 test statistic for two proportions, as is widely done in, e.g., epidemiology [Mie85]. Given a probe attribute $D \in P$ and two attributes A and B in R, the χ^2 test statistic is after some simplifications

$$E_{\chi^2}(A, B, D) = \frac{(fr(D, r_A) - fr(D, r_B))^2 fr(A, r) fr(B, r)(n-1)}{fr(D, r)(1 - fr(D, r))(fr(A, r) + fr(B, r))}$$

where n is the size of the relation r. When summed over all the probes D we get a distance measure $d_{E_{\chi^2},P}(A, B) = \sum_{D \in P} E_{\chi^2}(A, B, D)$. This measure is χ^2 distributed with $|P|$ degrees of freedom.

One might be tempted to use $d_{E_{\chi^2},P}$ or some similar notion as an external measure of similarity. Unfortunately, this measure suffers from the same problems as any other χ^2 based measure (see Section 3.2), and we need some other E_{f} measure. One such alternative is to define $E_{\mathsf{f}}(A, B, D)$ as the difference in the frequencies of the probe attribute D in the subrelations r_A and r_B. Then we have to following.

Definition 3.11 Let r_A and r_B be subrelations of r, P a set of probe attributes, and D a probe attribute in P. The difference in the frequencies of the probe attribute D in the relations r_A and r_B is

$$E_{fr}(A, B, D) = |\ fr(D, r_A) - fr(D, r_B)\ |.$$

Now the external distance between attributes A and B is

$$d_{E_{fr},P}(A, B) = \sum_{D \in P} E_{fr}(A, B, D).$$

Because $fr(D, r_A) = conf(A \Rightarrow D)$ and $fr(D, r_B) = conf(B \Rightarrow D)$, the measure $d_{E_{fr},P}$ can be also expressed as

$$d_{E_{conf},P}(A, B) = \sum_{D \in P} |\, conf(A \Rightarrow D) - conf(B \Rightarrow D)\,|. \qquad \square$$

The measure $d_{E_{fr},P}$ resembles the Manhattan distance [KR90, Nii87], and thus, it could be a metric. It is, however, only a pseudometric, because the value of it can be zero even if the attributes compared are not identical, i.e., $A \neq B$. This happens when $fr(D, r_A) = fr(D, r_B)$ with every probe attribute D. This is still not a problem because even in such a case the attributes A and B are similar: they are similar in the absence of the probe attributes. Note that for the internal distance $d_{I_{conf}}$ we have $d_{I_{conf}}(A, B) = d_{E_{fr},\{A,B\}}(A, B)$.

Example 3.20 Consider first products *milk* and *sausage* in the example market basket data. Assume then that we have a probe set $P = \{\text{beer}, \text{Coke}, \text{Pepsi}\}$. With this probe set, the products milk and sausage have the external distance

$$\begin{aligned} d_{E_{fr},P}(\text{milk}, \text{sausage}) &= E_{fr}(\text{milk}, \text{sausage}, \text{beer}) \\ &\quad + E_{fr}(\text{milk}, \text{sausage}, \text{Coke}) \\ &\quad + E_{fr}(\text{milk}, \text{sausage}, \text{Pepsi}) \\ &= |\, \tfrac{1}{6} - \tfrac{1}{6}\,| + |\, \tfrac{1}{6} - \tfrac{1}{6}\,| + |\, \tfrac{2}{6} - \tfrac{2}{6}\,| = 0. \end{aligned}$$

The same result can be also obtained with any non-empty subset of P. This result means that with respect to buying beer, Coke and Pepsi the customers buying milk and sausage behave similarly.

Consider then the products *chips* and *sausage* in the same relation. If we now have a probe set $P = \{\text{milk}\}$, these products have an external distance $d_{E_{fr},P}(\text{chips}, \text{sausage}) = |\, \frac{3}{6} - \frac{3}{6}\,| = 0$. Therefore, the products chips and sausage are similar regarding the product milk.

The external distance between the products *mustard* and *sausage* in the example market basket data becomes zero if we use a probe set $P = R \setminus \{\text{mustard}, \text{sausage}\}$. Also any subset of P or even an empty probe set gives the same result. This is due to the fact that mustard and sausage are substitute products. $\square$

In the previous example, all the three product pairs above were found similar with respect to some set of probes. These results are very different from the results obtained with the different internal measures in Example 3.17. Therefore, by using the external distance measure $d_{E_{fr},P}$ even such pairs of attributes that according to internal measures are determined totally different can be found to be highly similar.

Variations

Definition 3.11 for the measure $d_{E_{fr},P}$ is by no means the only possible one. We have at least the following three ways of defining an external distance between attributes A and B.

1. Instead of using a function resembling the Manhattan distance we could use a function corresponding to more general *Minkowski distance* [KR90, Nii87]. The external distance between attributes A and B would then be

$$d_{E_{fr},P}(A,B) = \left[\sum_{D \in P} |\, fr(D, r_A) - fr(D, r_B)\,|^p \right]^{1/p}.$$

2. We could give for each probe D a weight $w(D)$ which could, for example, describe its significance in the relation r. The external distance between attributes A and B with this way would be

$$d_{E_{fr},P}(A,B) = \sum_{D \in P} w(D) \cdot |\, fr(D, r_A) - fr(D, r_B)\,|.$$

3. The probe set P could be generalized so that it would be a set of boolean formulas θ_i on attributes. Then we would have the external measure

$$d_{E_{fr},P}(A,B) = \sum_{i} |\, fr(\theta_i, r_A) - fr(\theta_i, r_B)\,|.$$

Each of these variations certainly influences the distances. The first variation should not have a large effect on them. The behavior of the other two variations is not immediately obvious, and it is also unsure if the second variation using the weights of probe attribute is a metric or even a pseudometric. Evaluating the exact importance and effect of these variations is not considered in this thesis, and is, therefore, left for further study.

Constructing external measures from internal measures

Our basic external distance measure and its variations for similarity between attributes A and B are based on using frequencies of attributes and confidencies of association rules. Instead of frequencies and confidences, we could, however, use any function which describes the behavior of probe attributes in relation to the attributes A and B.

One set of such functions are the set of internal measures for attribute similarity. Given a probe set $P = \{D_1, D_2, \ldots, D_k\}$ and attributes A and B, an internal distance can be used to define an external distance between attributes

A and B as follows. Assume that internal distances d_{I_f} between the attribute A and all the probe attributes $D \in P$ are presented as a vector

$$v_{A,P} = [d_{I_f}(A, D_1), \ldots, d_{I_f}(A, D_k)].$$

Similarly, internal distances d_{I_f} between the attribute B and all the probe attributes D can be presented as $v_{B,P} = [d_{I_f}(B, D_1), \ldots, d_{I_f}(B, D_k)]$. Then, the external distance between the attributes A and B can be defined using any suitable distance notion d between the vectors $v_{A,P}$ and $v_{B,P}$.

Example 3.21 Consider products *chips* and *sausage* in the example market basket data. Assume that we have a probe set $P = \{\text{beer}, \text{Coke}, \text{Pepsi}\}$ and use the internal distance $d_{I_{sd}}$ for describing relations between the interesting products and the probes. Then for the product chips we have the vector $v_{\text{chips},P} = [0.91, 0.80, 0.91]$. Similarly, for the product sausage we have the vector $v_{\text{sausage},P} = [0.91, 0.80, 0.91]$. If we now use as the distance d a measure corresponding to the Manhattan distance, the external distance between the products chips and sausage is zero. Therefore, the customers buying chips and sausage are said to behave similarly with respect to buying beer, Coke and Pepsi. □

Selection of probe attributes

In developing the external measure of similarity our goal was that the probes describe the facets of the subrelations that the user thinks are important. Because the probe set defines the viewpoint from which similarity is judged, different selections produce different measures. This is explicitly shown by the experiments described in Section 3.5. Therefore, a proper selection of the probe set is crucial for the usefulness the external measure of attribute similarity.

It is clear that there is no single optimal solution to the probe selection problem. Ideally, the user should have sufficient domain knowledge to determine which attributes should be used as probes and which not. Even if we know that the problem of selecting probes is highly dependent on the application domain and the situation considered, we still try to describe some general strategies that can help the user in the selection of the probes.

The simplest way of choosing probes is, of course, to take to the probe set all the other attributes than the attributes A and B, i.e., use the set $P = R \setminus \{A, B\}$ as a probe set. This set is probably in most cases inappropriate, especially, if the number of attributes in the relation r is high. Another simple ways is to select as probes a fixed amount of attributes, for example, those with the highest or lowest frequencies. We could also define a threshold for the frequency of attributes and choose as probes the attributes whose frequency is higher than the given threshold.

When the number of attributes in the probe set P grows, the distance between attributes A and B, of course, increases. If we add one probe D to the probe set P, we get a new probe set $Q = P \cup \{D\}$. Using Definition 3.11, the distance between attributes A and B with the probe set Q is

$$d_{E_{fr},Q}(A,B) = d_{E_{fr},P}(A,B) + |\, conf(A \Rightarrow D) - conf(B \Rightarrow D)\,|.$$

Despite the number of probes in P, the external distance will always be less than the size of the probe set, i.e., $d_{E_{fr},P}(A,B) \leq |P|$.

The most frequent attributes tend to co-occur with almost every attribute in the relation r. If a probe D is such an attribute, the confidence of association rules $A \Rightarrow D$ and $B \Rightarrow D$ are both nearly 1. This means that the probe D has only a little effect on the whole distance. An extreme case is when $fr(D,r) = 1$, because then the confidences above are both exactly 1 and the external distance $d_{E_{fr},\{D\}}(A,B) = 0$. Thus, such a probe D has no effect at all on the external distance. If the frequency $fr(D,r)$ is, however, low compared to the frequencies of the attributes A and B, the confidences $conf(A \Rightarrow D)$ and $conf(B \Rightarrow D)$ are low, too. This means that also in this case the change in the external distance is small. Thus, adding or excluding an attribute with a very high or very low frequency does not typically produce dramatic changes in the distance value.

The probe selection problem can also be considered more formally. Assume, for example, that we for some reason know (or want) that attributes A and B are more similar that attributes A and C. Then we can try to search for a probe set that implies this fact (if one exists), and use this probe set to find distances between the other interesting attributes. The problem of finding such a probe set can be solved in different ways. For the first, we can search for all the single probes D satisfying $d_{E_{fr},P}(A,B) < d_{E_{fr},P}(A,C)$, or alternatively, the largest set of probes P satisfying the same condition. Similarly, if we know several such constraints on the similarities between attributes, we can search for all the single probes D satisfying all these constraints or all the possible probe sets P satisfying them. Sketches of algorithms for finding such probe attributes are given in [DMR97].

3.4 Algorithms for computing attribute similarity

Algorithm for internal distances

For computing the internal distances $d_{I_{sd}}$ and $d_{I_{conf}}$ between attributes we can use Algorithm 3.1. In implementation of the algorithm we just have to define which measure we want to use. This trivial algorithm gets as input the set $\mathcal{AI}$ of interesting attributes, the frequencies $fr(A)$ for each $A \in \mathcal{AI}$, and the frequencies $fr(AB)$ when both A and B are in the set $\mathcal{AI}$. In the case of the measure $d_{I_{conf}}$, we could also give as input the confidence of the association rules $A \Rightarrow B$ and $B \Rightarrow A$, instead of the frequencies. Output of the algorithm are the pairwise internal distances between the given attributes.

Algorithm 3.1 Internal distance d_{I_f} between attributes

Input: Interesting attributes $\mathcal{AI}$ and the frequencies of them and their pairwise combinations.
Output: Pairwise internal distances between the given attributes in $\mathcal{AI}$.
Method:

1. **for** all attribute pairs (A, B) where A and $B \in \mathcal{AI}$ **do**
2. calculate $d_{I_f}(A, B)$;
3. **od**;
4. output the matrix of the pairwise internal distances;

Algorithm for external distances

Computing the external distance $d_{E_{fr},P}$ between attributes can be done by using the algorithm 3.2. The algorithm gets as input the set $\mathcal{AI}$ of interesting attributes and the probe set P. Also the the frequencies $fr(A)$ of the interesting attributes, the frequencies $fr(D)$ of the probes and the frequencies $fr(AD)$, for each $A \in \mathcal{AI}$ and $D \in P$, are given to the algorithm. Another possibility would be to give as input, instead of the frequencies above, the frequencies of all probe attributes $D \in P$ in all the subrelations r_A. This is the same thing, if we would give as input the confidences of the rules $A \Rightarrow D$ for each $A \in \mathcal{AI}$ and $D \in P$. The algorithm computes first for each probe D the value of the function $E_{fr}(A, B, D)$ and then adds it to the distance value already computed. Output of the algorithm is the set of pairwise external distances between the given attributes.

Algorithm 3.2 External similarity $d_{E_{fr},P}$ between attributes

Input: A set of interesting attributes $\mathcal{AI}$, a probe set P, frequencies of the interesting attributes and frequencies of all (probe, interesting attribute) -pairs.
Output: Pairwise external distances between the given attributes in $\mathcal{AI}$.
Method:

1. **for** all attribute pairs (A, B) where A and $B \in \mathcal{AI}$ **do**
2. **for** all probes $D \in P$ **do**
3. calculate $E_{fr}(A, B, D)$;
4. add $E_{fr}(A, B, D)$ to $d_{E_{fr},P}$;
5. **od**;
6. **od**;
7. output the matrix of the pairwise external distances;

Complexity considerations

Computing the frequencies needed in the algorithms above is a special case of the problem of computing all the frequent sets that arises in association rule discovery [AIS93, AMS+96, Toi96]. The difference to association rule discovery is that in the case of attribute similarity we do not need all the frequent sets, just the frequencies of the sets containing interesting attributes and/or probe attributes. If we are not interested in probe attributes of small frequency, we can also use variations of the Apriori [AMS+96] algorithm for the computations needed. This method is fast and scales nicely to very large data sets.

For computing the internal measures $d_{I_{sd}}$ and $d_{I_{conf}}$ we need to have the frequencies of all attributes $A \in \mathcal{AI}$ and also the frequencies $fr(AB)$ when both A and B are in the set $\mathcal{AI}$. There are $|\mathcal{AI}| + \binom{|\mathcal{AI}|}{2}$ such frequencies. In the case of the measure $d_{I_{conf}}$ we would need $\binom{|\mathcal{AI}|}{2}$ confidence values. If there are totally n attributes in the set $\mathcal{AI}$, Algorithm 3.1 takes $O(n^2)$ space. Because the algorithm computes distances between $\binom{|\mathcal{AI}|}{2}$ pairs of attributes, the time complexity of the algorithm is also $O(n^2)$.

For computing the external distance we need to have the frequencies of all attributes $A \in \mathcal{AI}$ and the pairwise frequencies $fr(AD)$ where $A \in \mathcal{AI}$ and $D \in P$. There are $|\mathcal{AI}| + |\mathcal{AI}| \times |P|$ such frequencies. If we use as input the confidence values of the association rules $A \Rightarrow D$ for each A and D, the number of values needed is $|\mathcal{AI}| \times |P|$. If there are n interesting attributes and k probe attributes, the space complexity of Algorithm 3.2 is $O(n \cdot k)$. Also this algorithm computes distances between $\binom{|\mathcal{AI}|}{2}$ pairs of attributes, and the time complexity of the algorithm, therefore, is $O(n^2)$.

3.5 Experiments

In this section we present experiments that we made on similarity between attributes. In Subsection 3.5.1 we describe the data sets used in the experiments, and in Subsection 3.5.2 we discuss the results obtained. All the experiments were run on a PC with 233 MHz Pentium processor and 64 MB main memory under the Linux operating system.

3.5.1 Data sets

In our experiments on similarity between attributes we used two data sets: the Reuters-21578 collection of newswire articles [Lew97], and the course enrollment data of the Department of Computer Science at the University of Helsinki.

Documents and keywords

The Reuters-21578 collection consists of 21578 news articles from 1987. Most articles have a few keywords describing their contents. For example, an article with title "National average prices for farmer-owned reserve" has keywords grain, wheat, corn, barley, oat, sorghum and USA. There are altogether 445 different keywords which are divided in five categories: economic subjects, exchanges, organizations, people and places. A total of 1862 articles have no keywords at all, which means that in our experiments the size of the data set was considered to be 19716 rows. One of the articles in the Reuters-21578 data set has 29 keywords, and the average number of keywords per article is slightly over 2.

For our experiments we chose as the set of interesting attributes 14 countries: *Argentina, Brazil, Canada, China, Colombia, Ecuador, France, Japan, Mexico, Venezuela, United Kingdom, USA, USSR* and *West Germany* [1]. As probe sets, we used three sets of related keywords: *economic terms*, *international organizations* and a set of *mixed terms*. The exact probe sets are given in Figure 3.7.

Students and courses

In the course enrollment data there is information about 6966 students of the Department of Computer Science at the University of Helsinki. The data was collected from the year 1989 to the year 1996. Each row in this relation describes all the course enrollments made by a student. For example, one first year student has enrolled in the courses Introduction to Computing, Computer Systems Organization, and Programming (Pascal). The courses are divided into three classes: basic, intermediate and advanced level courses. The number of different courses in the data set is 173. About one third of the students (2528 students) has enrolled only to one course, and one student to a total of 33 courses. The average number of enrolled courses per student is close to 5.

In our experiments we used as the set of interesting attributes nine advanced level courses: *User Interfaces, Database Systems II, Object-Oriented Databases, String Processing, Design and Analysis of Algorithms, Neural Networks, Computer Networks, Compilers* and *Distributed Operating Systems*. Of these nine courses the first three belong to courses from the section of information systems, the next three to courses from the section of general orientation in computer science, and the last three to courses from the software section. In computing the external distances we used four probe sets: a set of *compulsory intermediate level courses*, a set of *optional intermediate level courses*, a *mixed set of advanced level courses*, and a set of *advanced courses from computer software section*. The exact probe sets are given in Figure 3.8. Each probe set, except the last one, contains at least one course from each of the three sections.

[1]The data set was collected in 1987, before the split of the USSR and the unification of Germany.

Probe set	Probe attributes
economic terms	earn, trade, interest
organizations	ec, opec, worldbank, oecd
mixed terms	acq, corn, crude, earn, grain, interest money-fx, rice, ship, trade, wheat

Figure 3.7: The probe sets of the Reuters-21578 data used in the experiments.

Probe set	Probe attributes
compulsory intermediate level courses	Computers and Operating Systems, Data Structures, Database Systems I, Theory of Computation
optional intermediate level courses	Computer Graphics, Data Communications, Computer-Aided Instruction
mixed set of advanced courses	Knowledge Bases, Logic Programming, Machine Learning
advanced courses from software section	Data Communications, Unix Data Communications Unix Platform

Figure 3.8: The probe sets of the course enrollment data used in the experiments.

3.5.2 Results and discussion

We started our experiments by comparing the internal distances given by the two measures $d_{I_{sd}}$ and $d_{I_{conf}}$ with the external distances given by the measure measure $d_{fr,P}$ for different probe sets P.

The actual values of a distance function are, obviously, irrelevant. We can multiply or divide the distance values by any constant without modifying the properties of the measure. What actually matters in many applications is only the relative order of the values. That is, as long as for all attributes A, B, and C we have $d(A,B) < d(A,C)$ if and only if $d'(A,B) < d'(A,C)$, the measures d and d' behave in the same way.

The top left plot in Figure 3.9 shows the distribution of points $(d_{I_{sd}}(A,B), d_{I_{conf}}(A,B))$ for all attribute pairs (A,B) of the 14 countries in the Reuters-21578 data set. The values of the $d_{I_{conf}}$ measure are all quite near 2, which indicates that the confidences of the rules $A \Rightarrow B$ and $B \Rightarrow A$ are both low. Similarly, a large fraction of the values of the $d_{I_{sd}}$ measure are close to 1. These results were as expected, because few pairs of the countries occur in the same articles.

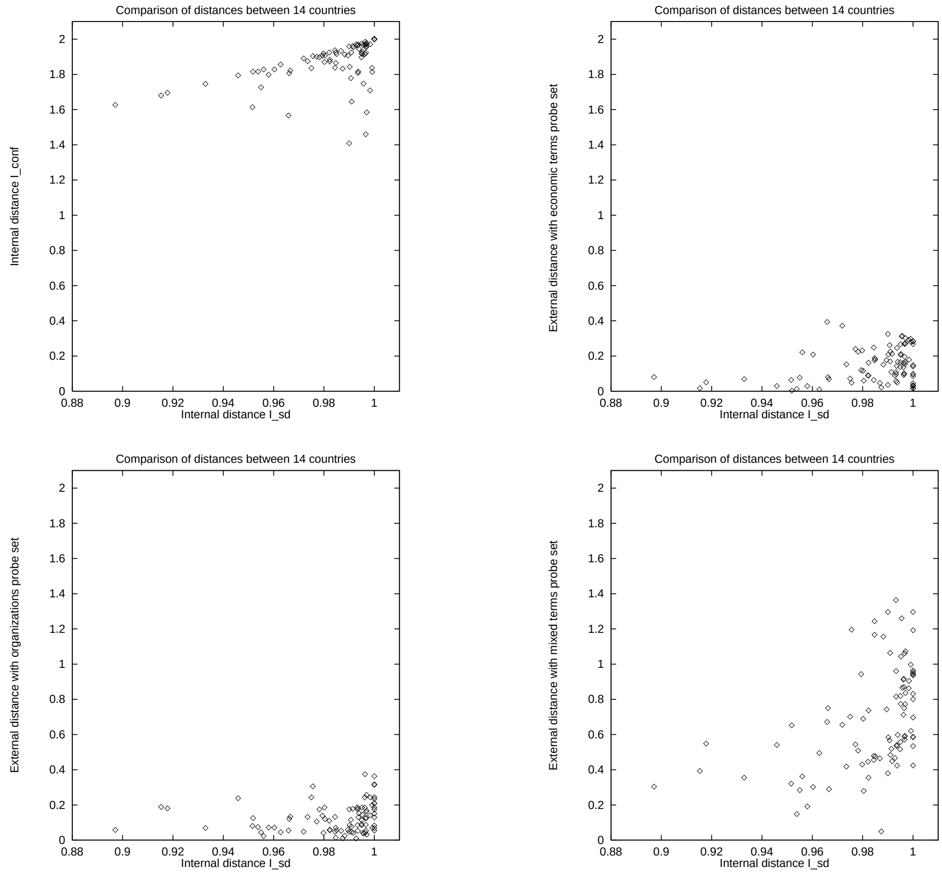

Figure 3.9: Relationships between internal distances and external distances between the 14 countries in the Reuters data.

The other three plots in Figure 3.9 describe how the internal distance given by the measure $d_{I_{sd}}$ are related to the external distances given by the measure $d_{fr}P$ for the three probe sets P in the Reuters-21378 data. The distributions of the points in these plots are fairly wide, especially the distribution of the value pairs of $d_{I_{sd}}$ and $d_{fr}P$ with the mixed term probe set. These results indicate that the internal and external measures truly measure different things.

Similar experiments were also made with the nine advanced level courses in the course enrollment data set. The top left plot in Figure 3.10 presents the distribution of points $(d_{I_{sd}}(A, B), d_{I_{conf}}(A, B))$ and the other four plots the distributions of points $(d_{I_{sd}}(A, B), d_{E_{fr}, P}(A, B))$ with the four different probe set. The values of the $d_{I_{sd}}$ measure and the $d_{I_{conf}}$ measure are rather high, indicating that the pairs of courses do not occur all too often on the same rows in the data set. The distributions of the points $(d_{I_{sd}}(A, B), d_{E_{fr}, P}(A, B))$ in the other three plots show once again that the internal and external measures

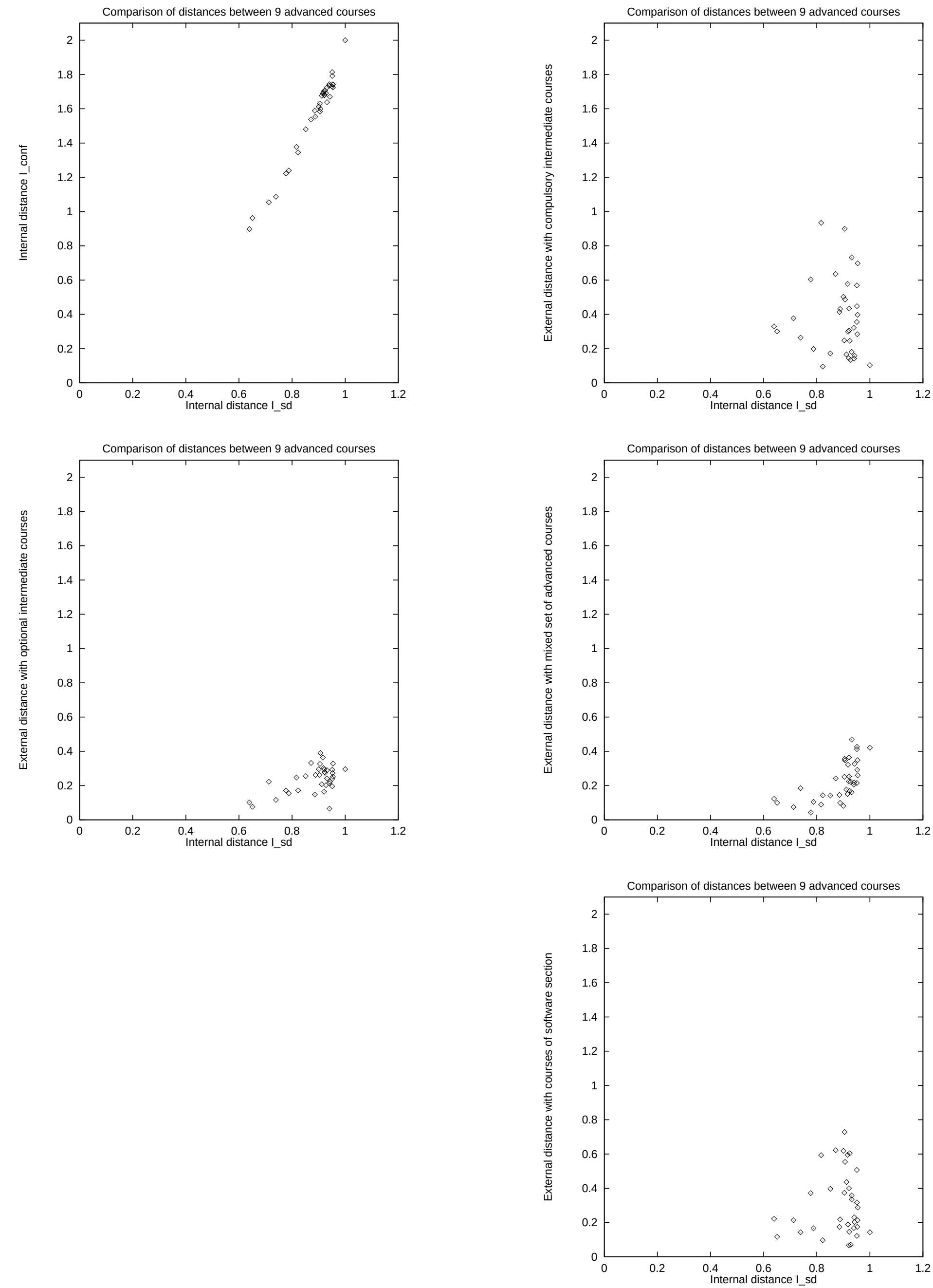

Figure 3.10: Relationships between internal distances and external distances between the nine advanced level courses in the course enrollment data.

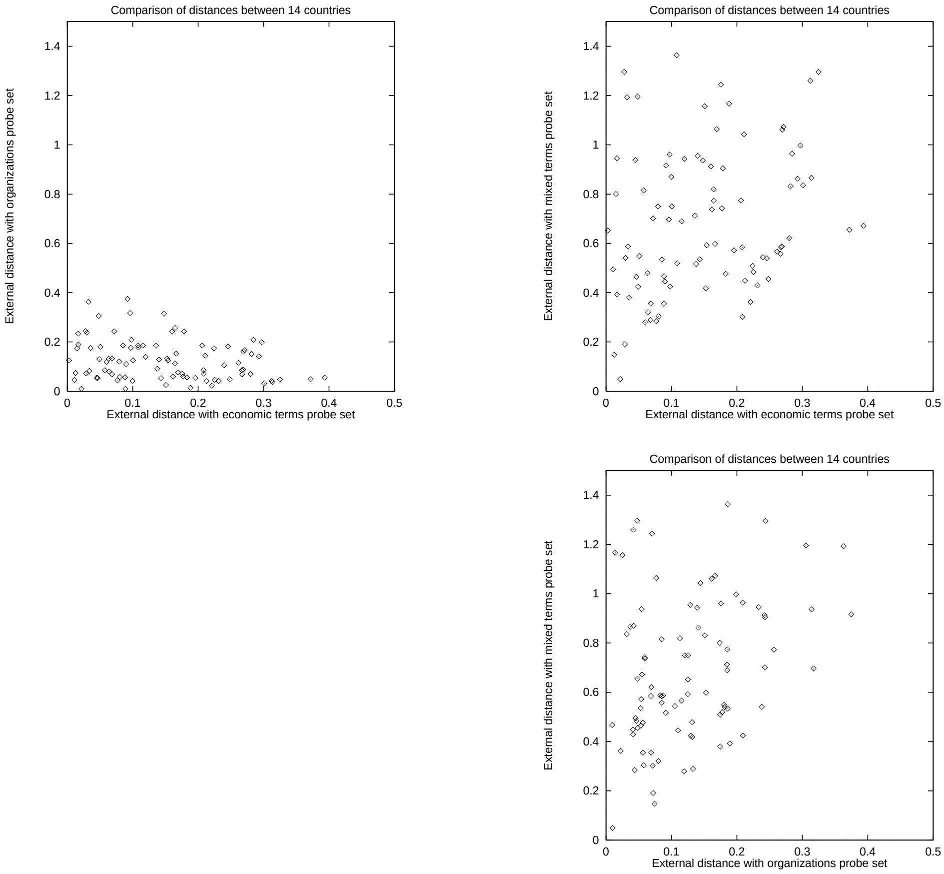

Figure 3.11: Relationships between external distances with various probe sets between the 14 countries in the Reuters data.

describe different aspects of the data. Note that in all these three cases the external measure states that the pairs of attributes are more similar to each other than the internal measure $d_{I_{sd}}$.

The main idea in constructing external measures between attributes was to let the choice of the probes affect the distances. Therefore, given two probe sets P and Q which have no relation to each other, we have no reason to assume that the measures $d_{E_{fr},P}$ and $d_{E_{fr},Q}$ should have any specific relationship. This is also shown by our experiments.

In Figure 3.11 there are plots comparing the distances given by the different external measures in the Reuters-21578 data, and in Figure 3.12 we have similar plots comparing the distances in the course enrollment data. The points in the plots are widely distributed, which indicated clearly that different probe sets produce different similarity notions. This is as expected: the probe set defines the point of view from which similarity between attributes is judged.

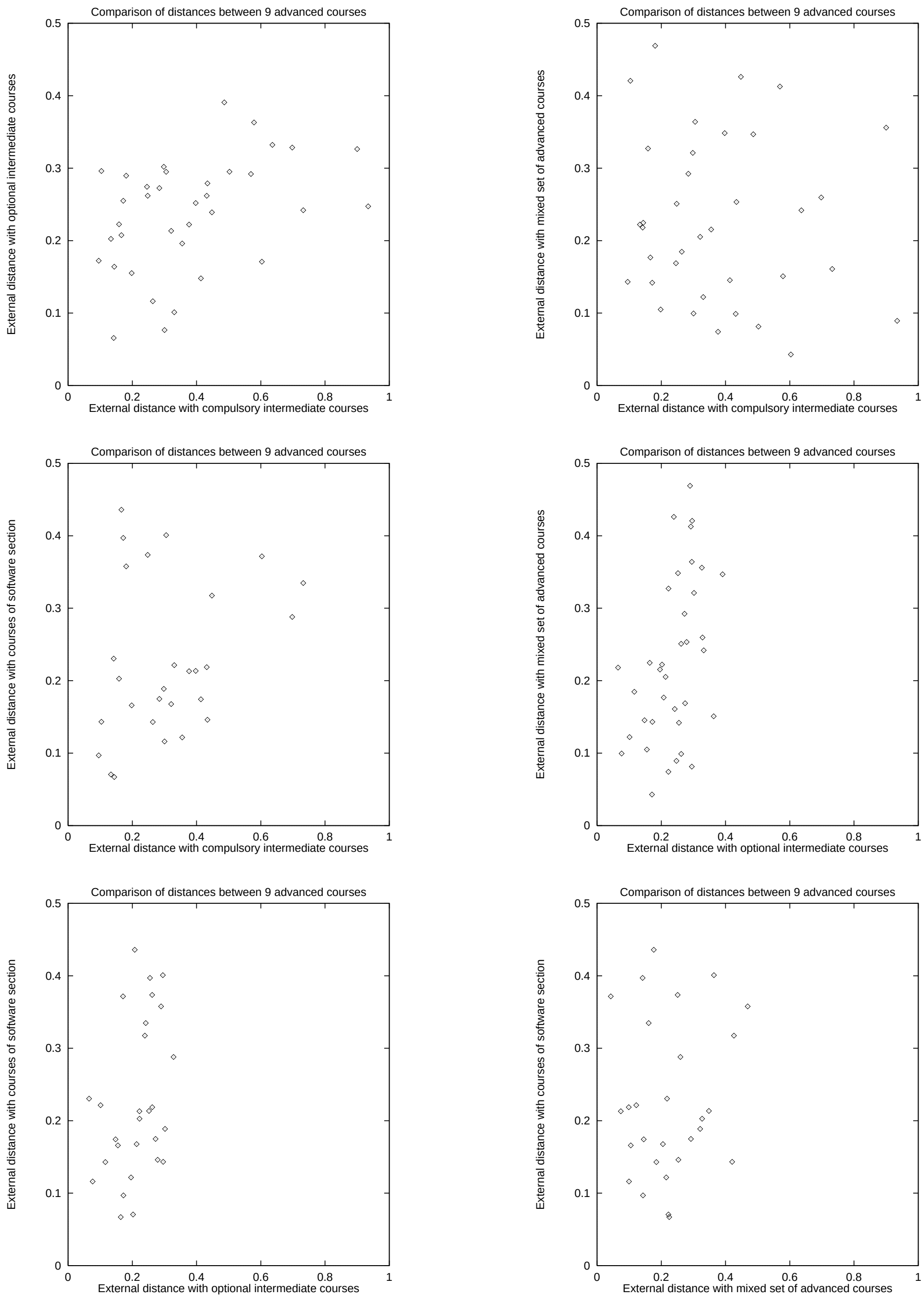

Figure 3.12: Relationships between external distances with different probe sets between the nine advanced level courses in the course enrollment data.

4 Similarity between event sequences

In this chapter we move to discuss how similarity between event sequences could be defined. We start by describing properties of event sequences in Section 4.1. Measures for similarity between event sequences are presented in Section 4.2. After that, in Section 4.3 we give the algorithms for computing the similarities. Experimental results are represented in Section 4.4. A part of the material in this chapter is based on [MR97].

4.1 Event sequences

Many times when we are using data mining techniques we are considering unordered data sets. Still, in many important application areas the data has a clear sequential structure. For example, in user interface studies, in WWW page request monitoring, or in telecommunication network monitoring a lot of data can easily be collected about the behavior of the user or the system. Formally, such data can be viewed as an event sequence.

Definition 4.1 Let $R = \{A_1, \ldots, A_m\}$ be a set of *event attributes* with domains $Dom(A_1), \ldots, Dom(A_m)$. Then an *event* is a $(m+1)$-tuple $(a_1, \ldots, a_m, t)$, where $a_i \in Dom(A_i)$ and t is a real number, the *occurrence time* of the event.

An *event sequence* is a collection of events over $R \cup \{T\}$, where the domain of the attribute T is the set of real numbers $\mathbb{R}$. The events in the event sequence are ordered ascending by their occurrence times t. □

In the examples of this chapter we use mainly artificial data but in a few cases also *telecommunication alarm data* and a *log of WWW page requests.*

Example 4.1 Telecommunication networks produce daily large amounts of alarms. An alarm is generated by a network element if it has detected some abnormal situation. Such an alarm flow can be viewed as an event sequence. Each alarm has several attributes like *module* and *severity*, indicating the element that sent the alarm, and the severity of the alarm, respectively. An alarm has also a type and an occurrence time associated to it. An example of a real alarm is

(2730, 30896, 19.8.1994 08:33:59, 2, Configuration of BCF failed)

where the attributes are the type of the alarm, the sending network element, the occurrence time of the alarm, the severity class, and a text describing the failure. □

Example 4.2 Page requests made in World Wide Web are often collected into a log. Event attributes of a page request are, e.g., the requested WWW *page*, the name of the *host* that made the request and the occurrence time of the request. An example of a page request is

(athene.cs.uni-magdeburg.de, 07/Aug/1996 15:37:11,
/mannila/data-mining-publications.html, 200, 12134)

where the first attribute is the requesting host, the second the time of the request, and the third the requested page. The last two attributes of the request describe the success status of the request. □

The number of attributes associated with each event can be high. Some of these event attributes can contain redundant or irrelevant information. In telecommunication alarm data, for example, alarm text can be considered to be redundant if it can be determined from the alarm type. On the other hand, the physical location of the network element sending the alarm can be, for example, considered irrelevant. This means that only few of the event attributes are really interesting when studying the similarity between event sequences. Therefore, we consider a simpler model of events where each event has only a type and an occurrence time.

Definition 4.2 Given a set $\mathcal{E}$ of event types, an event is a pair (e, t) where $e \in \mathcal{E}$ is an event type and $t \in \mathbb{R}$ is the occurrence time of the event. Then an event sequence $\mathcal{S}$ is an ordered collection of events, i.e.,

$$\mathcal{S} = \langle (e_1, t_1), (e_2, t_2), \ldots, (e_n, t_n) \rangle,$$

where $e_i \in \mathcal{E}$, and $t_i \leq t_{i+1}$ for all $i = 1, \ldots, n-1$. The length of the sequence $\mathcal{S}$ is $|\mathcal{S}| = n$. A sequence that consists only of event types in time order, i.e.,

$$S = \langle e_1, e_2, \ldots, e_n \rangle,$$

where each $e_i \in \mathcal{E}$, is called an *event type sequence.* □

Example 4.3 Assume the event type set $\mathcal{E} = \{A, B, C, D, E, F, G\}$. An example of an event sequence consisting of events $e \in \mathcal{E}$ is represented in Figure 4.1. Formally, this sequence can be expressed as

$$\mathcal{S} = \langle (A, 30), (E, 31), (D, 32), (A, 35), (B, 37), \ldots, (A, 64) \rangle.$$

An event type sequence corresponding to this event sequence is

$$S = \langle A, E, D, A, B, \ldots, A \rangle.$$

Both the event sequence $\mathcal{S}$ and the event type sequence S contain 21 events. We would get even longer sequences if the events whose occurrence times are less than 30 or more than 65 were taken into account. □

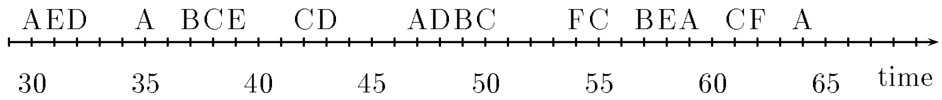

Figure 4.1: An event sequence on the time axis.

Example 4.4 An example of an event sequence in telecommunication data is an alarm sequence

$$\begin{aligned}\mathcal{S}_{alarm} = \; & \langle (1903, 5), (7402, 5), (7172, 14), (7310, 16), (7172, 17), \\ & (7177, 18), (7010, 28), (7002, 36), (7177, 52), (7177, 57)\rangle\end{aligned}$$

where the numbers like 1903, 7402 and 7172 represent types of the alarms. The event type sequence corresponding to the alarm sequence $\mathcal{S}_{alarm}$ is

$$S_{alarm} = \; \langle 1903, 7402, 7172, 7310, 7172, 7177, 7010, 7002, 7177, 7177\rangle.$$

One problem with analyzing the real alarm data is related to the occurrence times of the alarms in the network. In the sequence $\mathcal{S}_{alarm}$ we have, for instance, alarms $(1903, 5)$ and $(7402, 5)$ in that order. This is not necessarily the real order of the alarms: they may have occured in an opposite order. Therefore, in analyzing the sequences we should consider both possibilities. Another problem associated with the occurrence times is that there can be differences of several minutes in the synchronization of the clocks in the network elements. This means that two alarms could have occurred exactly at the same time but still have different occurrence times in the given sequence. Despite these known problems with occurrence times, we consider in this thesis events of the sequences strictly in the given order, and leave handling of those problems for future study. □

Example 4.5 Assume that the names of the requested WWW pages are considered to be the event types. Then an example of a short event sequence in WWW log data is a sequence of (requested page,time) -pairs

$$\begin{aligned}\mathcal{S}_{WWW} = \; & \langle(\text{/research/pmdm/datamining/,15}), (\text{/mannila/cv.ps,137}), \\ & (\text{/mannila/data-mining-publications.html,201}), (\text{/mannila/,211})\rangle.\end{aligned}$$

The event type sequence corresponding to the sequence $\mathcal{S}_{WWW}$ is

$$\begin{aligned}S_{WWW} = \; & \langle \text{/research/pmdm/datamining/, /mannila/cv.ps,} \\ & \text{/mannila/data-mining-publications.html, /mannila/}\rangle.\end{aligned}$$ □

The real event sequences are usually extremely long, and they are difficult to analyze as such. Therefore, we need a way of selecting shorter sequences suitable for our purposes. This leads us to the definition of event subsequence.

Definition 4.3 Let $\mathcal{S}$ be an event sequence. A boolean expression θ is called a *selection condition* on the events of the sequence $\mathcal{S}$. An *event subsequence* of the sequence $\mathcal{S}$ is an event sequence that satisfies θ, i.e.,

$$\mathcal{S}(\theta) = \langle (e_i, t_i) \,|\, (e_i, t_i) \in \mathcal{S} \text{ and it satisfies } \theta \,\rangle.$$

Similarly, an event type sequence that satisfies θ, i.e.,

$$S(\theta) = \langle e_i \,|\, e_i \in S \text{ and it satisfies } \theta \,\rangle.$$

is called an *event type subsequence*. □

The definition of an event subsequence $\mathcal{S}(\theta)$ (similarly the definition of an event type subsequence) resembles the definition of a subrelation r_θ in Section 3.1. The form of a selection condition θ in Definition 4.3, however, was left quite open. The condition θ can contain restrictions on event types, occurrence times or both of them. In the case of event type sequences, of course, only restrictions on event types are meaningful. Simple constraints on event types and occurrence times can be combined with different boolean operators.

The simplest constraints on event types are of the form "$e_i = A$". Two examples of more complex selection conditions θ on event types are "$e_i = A \vee e_i = B$" and "$e_i = B \wedge e_{i+1} = C$". Also the number of the events in the subsequence could be restricted. Constraints on occurrence times, on the other hand, can select into the subsequence all the events in a given time period before or after a given type of an event. They could also influence the time difference between the first and the last event in the subsequence.

Note that a subsequence does not have to be a continguous part of the original sequence; it just has to maintain the order of the events. Hence, a subsequence is a subset of the events of the original sequence in their "relative" order.

Example 4.6 Consider the event sequence in Figure 4.1. From it we can, for example, select a subsequence

$$\mathcal{S}(e_i = A) = \langle (A, 30),\ (A, 35),\ (A, 47),\ (A, 59),\ (A, 64) \rangle$$

which is a sequence of the events of type A. □

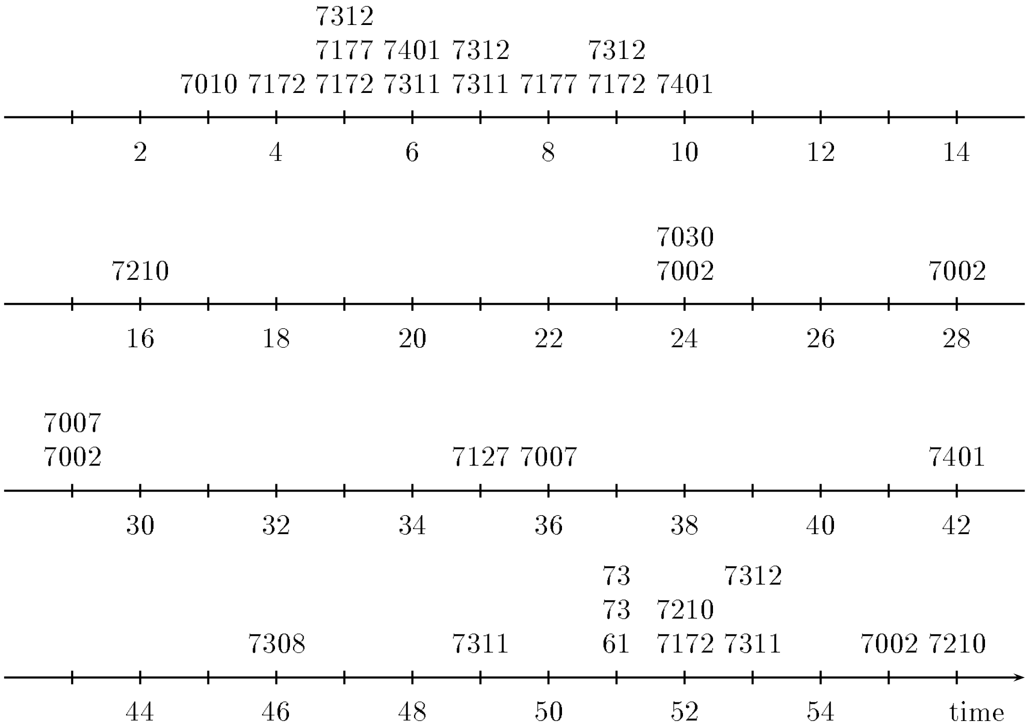

Figure 4.2: An example alarm sequence on the time axis.

Example 4.7 Consider the alarm sequence in Figure 4.2. From it we can extract, for example, the following subsequences. A subsequence of events whose type is 7172 consists of four events, i.e.,

$$\mathcal{S}(e = 7172) = \langle (7172, 4), (7172, 5), (7172, 9), (7172, 52) \rangle.$$

Assume that we are interested in what happens, for instance, at most two seconds before that an event of type 7210 occurs, i.e., we want to find subsequences $\mathcal{S}(e_j = 7210 \wedge t_i \geq t_j - 2 \wedge i < j)$. In the example sequence there are three events of type 7210, at times 16, 52 and 56. Therefore, we find three subsequences

$$\begin{aligned} &\mathcal{S}_1 = \langle \rangle, \\ &\mathcal{S}_2 = \langle (61, 51), (73, 51), (73, 51), (7172, 52) \rangle, \text{ and} \\ &\mathcal{S}_3 = \langle (7002, 55) \rangle. \end{aligned}$$

The subsequence preceding the first occurrence of the 7210-type event is empty because there has not occurred any events two seconds before that event.

In the case of our example alarm sequence we could also search for subsequences where we give some range to the event types. We could, for instance, be interested in a subsequence where the types of all the events are between 7000 and 7010, i.e., a subsequence $\mathcal{S}(7000 \leq e_i \leq 7010)$. The resulting subsequence consists of seven events and is $\langle(7010,3),(7002,24),(7002,28),(7002,29),(7007,29),(7007,36),(7002,55)\rangle$. $\square$

In recent years, interest in knowledge discovery from event sequences has been quite intensive; see, for example, [Lai93, MTV95, MT96, OC96, Toi96]. Also the problem of similarity between sequences is considered in many areas such as text databases, genetics, biomedical measurements, telecommunication network measurements, economic databases, and scientific databases. This research, however, concentrates on sequences of numerical values. Especially time series and similarity queries on them have been studied widely; see, for example, [AFS93, APWZ95, ALSS95, DGM97, FRM93, JMM95, KJF97, RM97, SHJM96]. Some of these consider similarity between long sequences, and some of them just finding similar subsequences. As far as we know no one before us has considered the similarity between event sequences.

4.2 Similarity measures

We consider similarity between both sequences of (event type,time) -pairs and event type sequences, i.e., sequences with and without occurrence times. We define similarity between two event sequences with a complementary notion of distance.

Definition 4.4 Given a set of event types $\mathcal{E}$ and a class of all sequences $\mathbf{S}$, a *distance* function d between event sequences is defined as $d : \mathbf{S} \to \mathbb{R}$. Given two event sequences $\mathcal{S}_1 \in \mathbf{S}$ and $\mathcal{S}_2 \in \mathbf{S}$, the distance between these two sequences is denoted by $d(\mathcal{S}_1, \mathcal{S}_2)$. $\square$

There are, of course, several ways of defining the distance between event sequences. The intuitive idea behind our distance measure is that it should reflect the amount of work needed to transform a sequence to another. The idea is formalized as *edit distance* which is a common and simple formalization of a distance between strings or sequences widely used in the analysis of textual strings [CR94, Ste94] and in the comparison of biological sequences [Gus97, SM97]. Modifications of edit distance have also been suggested as a similarity measure between numerical sequences [BYÖ97, BÖ97, YÖ96] and behavioral sequences [LB97].

Definition 4.5 Let $\mathcal{O}$ be a set of *edit operations* allowed in the transformation of sequences and o_i an edit operation in $\mathcal{O}$. A transformation between sequences can be presented by giving a series of needed edit operations, called as an *operation sequence.* An operation sequence of k edit operations is denoted as $O = \langle o_1, o_2, \ldots, o_k \rangle$.

An alternate way of presenting the transformation is to give an explicit *alignment* of the two sequences. An alignment of two sequences is obtained by first adding chosen spaces (or dashes) into the sequences and then placing the two resulting sequences one above the other so that every event or space in either sequence is matched with an unique event or space in the other sequence. □

The set of edit operations depends on the type of the sequences considered. For example, in traditional text string matching and biosequence comparison an insertion, a deletion and a substitution of characters form a standard edit operation set. Whatever edit operations are chosen, a non-operation called match is always used in computing the edit distance.

Example 4.8 Let $\mathcal{E} = \{A, B, C, D, E\}$ an event type set and $\mathcal{O} = \{insert, delete\}$ a set of edit operations. An event type sequence $S_1 = \langle A, B, A, C, B, D \rangle$ can transformed to an event type sequence $S_2 = \langle A, B, C, C, A, D \rangle$, for instance, with an operation sequence

$$\begin{array}{ll} \langle A, B, A, C, B, D \rangle & \text{delete A} \\ \langle A, B, C, B, D \rangle & \text{insert C} \\ \langle A, B, C, C, B, D \rangle & \text{insert A} \\ \langle A, B, C, C, A, B, D \rangle & \text{delete B} \\ \langle A, B, C, C, A, D \rangle. & \end{array}$$

This transformation can also be presented as an alignment

$$\begin{array}{llllllll} \langle\, \mathrm{A} & \mathrm{B} & \mathrm{A} & \mathrm{C} & \text{-} & \text{-} & \mathrm{B} & \mathrm{D}\rangle \\ \langle\, \mathrm{A} & \mathrm{B} & \text{-} & \mathrm{C} & \mathrm{C} & \mathrm{A} & \text{-} & \mathrm{D}\rangle. \end{array}$$

In this alignment, events A and B match their counterparts, an event A is deleted (opposite a space), an event C matches its counterpart, events C and A are inserted, an event B deleted, and an event D matches its counterpart. This alignment corresponds to the operation sequence above. □

In traditional text matching and comparison of biosequences, the edit distance between two strings or sequences is often defined as the the minimum number of edit operations needed to transform a string (sequence) to another. This sort of an edit distance is called the *Levenshtein distance* [CR94, Ste94, Gus97]. We want, however, the measure of distance between event sequences to be more general.

Definition 4.6 Let $\mathcal{O}$ be a set of edit operations o_i. To each operation o_i we associate a *cost*, denoted by $\mathsf{c}(o_i)$. A cost of an operation sequence O is a sum of the costs of individual operations, i.e.,

$$\mathsf{c}(O) = \sum_{i=1}^{k} \mathsf{c}(o_i).$$

An *optimal operation sequence* $\hat{O}$ is the operation sequence with the minimum cost. □

Example 4.9 Assume that all the operations $o \in \mathcal{O}$ have constant *unit costs* for operations, i.e., for each operation we have $\mathsf{c}(o) = 1$. If we have a set $\mathcal{O} = \{insert, delete, substitute\}$ of operations where *substitute* means replacing an event with another, the unit costs of operations are

$$\begin{aligned}
&\mathsf{c}(insert(e)) = 1\\
&\mathsf{c}(delete(e)) = 1\\
&\mathsf{c}(substitute(e,f)) = \begin{cases} 0 & \text{if } e = f\\ 1 & \text{if } e \neq f. \end{cases}
\end{aligned}$$

Now the cost of an operation sequence is always the number of the operations needed in the transformation of one event type sequence to another. Further, the edit distance between two event type sequences is the minimum of the costs of all the possible operation sequences, and, therefore, the measure we use for the distance between event type sequences is the Levenshtein distance.

Another possibility could be to use edit operation costs

$$\begin{aligned}
&\mathsf{c}(insert(e)) = 1\\
&\mathsf{c}(delete(e)) = 1\\
&\mathsf{c}(substitute(e,f)) = \begin{cases} 0 & \text{if } e = f\\ 2 & \text{if } e \neq f. \end{cases}
\end{aligned}$$

In this case, the cost of substituting an event with another is as much as first deleting an event and then inserting the other to the sequence. In traditional string matching this kind of costs are used in searching the longest common subsequence of the two strings. □

Now we are ready to give a formal definition for the distance between two event sequences.

Definition 4.7 Let $\mathcal{S}_1 = \langle (e_1,t_1),(e_2,t_2),(e_3,t_3),\ldots,(e_m,t_m)\rangle$ and $\mathcal{S}_2 = \langle (f_1,u_1),(f_2,u_2),(f_3,u_3),\ldots,(f_n,u_n)\rangle$ be two event sequences. The edit distance $d(\mathcal{S}_1,\mathcal{S}_2)$ is defined as

$$\begin{aligned}
d(\mathcal{S}_1,\mathcal{S}_2) \;=\; & min\ \{\ \mathsf{c}(O_j) \mid\\
& O_j \text{ is an operation sequence transforming}\\
& \text{sequence } \mathcal{S}_1 \text{ to sequence } \mathcal{S}_2\ \}.
\end{aligned}$$

That is, the cost of the optimal operation sequence $\hat{O}$ transforming sequence $\mathcal{S}_1$ to sequence $\mathcal{S}_2$. The definition holds also for the distance between two event type sequences. □

If the edit distance between two event sequences (event type sequences) is computed according to Definitions 4.6 and 4.7 we are actually considering a *weighted edit distance*. If each operation in the operation set has an arbitrary weight, we talk about an *operation-weight edit distance* which is a special case of an *alphabet-weight edit distance* [Gus97]. In this last case the costs of operations are dependent on the type of the event considered.

Example 4.10 Consider again the two event type sequences S_1 and S_2 in Example 4.8 and an operation set $\mathcal{O} = \{insert, delete, substitute\}$. We have, for example, the following three alignments of the sequences:

$$1. \quad \begin{array}{llllllll} \langle\, \mathrm{A} & \mathrm{B} & \mathrm{A} & \mathrm{C} & - & - & \mathrm{B} & \mathrm{D} \rangle \\ \langle\, \mathrm{A} & \mathrm{B} & - & \mathrm{C} & \mathrm{C} & \mathrm{A} & - & \mathrm{D} \rangle \end{array}$$

$$2. \quad \begin{array}{lllllll} \langle\, \mathrm{A} & \mathrm{B} & \mathrm{A} & \mathrm{C} & - & \mathrm{B} & \mathrm{D} \rangle \\ \langle\, \mathrm{A} & \mathrm{B} & \mathrm{C} & \mathrm{C} & \mathrm{A} & - & \mathrm{D} \rangle \end{array}$$

$$3. \quad \begin{array}{llllll} \langle\, \mathrm{A} & \mathrm{B} & \mathrm{A} & \mathrm{C} & \mathrm{B} & \mathrm{D} \rangle \\ \langle\, \mathrm{A} & \mathrm{B} & \mathrm{C} & \mathrm{C} & \mathrm{A} & \mathrm{D} \rangle \end{array}$$

If operations have constant unit costs, i.e.,

$$\begin{array}{l} \mathsf{c}\,(insert(e)) = 1 \\ \mathsf{c}\,(delete(e)) = 1 \\ \mathsf{c}\,(substitute(e,f)) = \left\{ \begin{array}{ll} 0 & \text{if } e = f \\ 1 & \text{if } e \neq f. \end{array} \right. \end{array}$$

the costs of the corresponding operation sequences are $d_1(S_1, S_2) = 4$, $d_2(S_1, S_2) = 3$, and $d_3(S_1, S_2) = 2$. The third operation sequence has the minimum cost, so it is the optimal operation sequence $\hat{O}$, and, therefore, the edit distance between the sequences is $d(S_1, S_2) = 2$. This is the same result as if we would have defined edit distance as the minimum number of operations needed in the transformation. □

The edit distance defined as the minimum number of operations needed for the transformation of sequences is known to be a metric. Problem with alpha-weight edit distance is the triangle inequality may not necessarily be satisfied, and, therefore, the measure is not necessarily a metric [Ste94]. This does not, however, prevent the use of weighted edit distance as a distance measure of event sequences.

4.2.1 Event type sequences

We consider first similarity between event type sequences. In this case, it is important to notice that we are talking about ordinary string matching.

Edit operations

In the case of event type sequences the set $\mathcal{O}$ consists of two parametrized edit operations: an insertion $\texttt{Ins}\,(e)$ that inserts an event of type e to a sequence, and a deletion $\texttt{Del}\,(e)$ that deletes an event of type e from a sequence. Intuitively, two event type sequences are similar if a lot of the events in them match and the optimal operation sequence $\hat{O}$ contains only few insertions and deletions.

Costs of operations

The costs of edit operations for event type sequences can be defined in different ways. The simplest approach is to use constant unit costs for each edit operation, i.e., the cost of an insertion is $\texttt{c}\,(\texttt{Ins}\,(e)) = 1$ and the cost of a deletion of an event is the same. The edit distance between event type sequences S_1 and S_2 is then the minimum number of operations needed to transform one sequence to the other sequence, the Levenshtein distance. This kind of costs are useful, for example, if we want to find the longest common subsequence of the two event type sequences S_1 and S_2.

Another possibility is to use alphabet-weighted costs, i.e., costs that are dependent on the type of the event inserted or deleted. We may, for example, want that the cost of adding (deleting) a rare event is higher than the cost of adding (deleting) a common event. In such a case the cost of an Ins-operation can be defined as

$$\texttt{c}\,(\texttt{Ins}\,(e)) = w(e),$$

where $w(e)$ is a constant proportional to $occ(e)^{-1}$, when $occ(e)$ is the number of occurrences of a e-type event in a long reference sequence from the same application area. Similarly, the cost of a deletion operation can be defined as $\texttt{c}\,(\texttt{Del}\,(e)) = w(e)$.

If two event type sequences S_1 and S_2 have no events of similar type, i.e., there are no events $e_i \in S_1$ and $f_j \in S_2$ such that $e_i = f_j$, the distance between these two event type sequences is

$$\sum_{e_i \in S_1} \texttt{c}\,(\texttt{Del}\,(e_i)) + \sum_{f_j \in S_2} \texttt{c}\,(\texttt{Ins}\,(f_j)).$$

The other extreme distance value, $d(S_1, S_2) = 0$, is reached, when all the events in the first sequence match in right order the events in the second sequence, i.e., $e_i = f_i$ for all i. This means also that the sequences have to be equally long.

Example 4.11 Let $\mathcal{E} = \{A, B, C, D, E\}$ be a set of event types and S_1 and S_2 the two event type sequences in Example 4.8. With constant unit costs the optimal operation sequence $\hat{O}$ transforming the sequence S_1 to the sequence S_2 is

$$\texttt{c}(\texttt{Del}(A)),\ \texttt{c}(\texttt{Ins}(C)),\ \texttt{c}(\texttt{Ins}(A)),\ \texttt{c}(\texttt{Del}(C))$$

and, therefore, the edit distance between these sequences is

$$\begin{aligned} d(S_1, S_2) &= \texttt{c}(\texttt{Del}(A)) + \texttt{c}(\texttt{Ins}(C)) + \texttt{c}(\texttt{Ins}(A)) + \texttt{c}(\texttt{Del}(C)) \\ &= 1 + 1 + 1 + 1 = 4. \end{aligned}$$

Suppose then that we have a reference sequence from which we get the following numbers of occurrences of different event types and, therefore, the costs of edit operations are the following.

e	$occ(e)$	$\texttt{c}(\texttt{Ins}(e)) = \texttt{c}(\texttt{Del}(e))$
A	100	0.0100
B	50	0.0200
C	20	0.0500
D	80	0.0125
E	10	0.1000

With these alphabet-weighted costs, the optimal operation sequence for the two event type sequences is the same as with unit costs and their edit distance is

$$d(S_1, S_2) = 0.01 + 0.05 + 0.01 + 0.02 = 0.09.$$

Assume then that we have two other event type sequences $S_3 = \langle A, B, C \rangle$ and $S_4 = \langle D, E, D \rangle$. These two sequences have no events of a similar type. Therefore, the optimal operation sequence $\hat{O}$ for these sequences is

$$\texttt{c}(\texttt{Del}(A)),\ \texttt{c}(\texttt{Del}(B)),\ \texttt{c}(\texttt{Del}(C)),\ \texttt{c}(\texttt{Ins}(D)),\ \texttt{c}(\texttt{Ins}(E)),\ \texttt{c}(\texttt{Ins}(D))$$

Now the edit distance between these event type sequences with unit operation costs is

$$\begin{aligned} d(S_3, S_4) &= \texttt{c}(\texttt{Del}(A)) + \texttt{c}(\texttt{Del}(B)) + \texttt{c}(\texttt{Del}(C)) \\ &\quad + \texttt{c}(\texttt{Ins}(D)) + \texttt{c}(\texttt{Ins}(E)) + \texttt{c}(\texttt{Ins}(D)) \\ &= 1 + 1 + 1 + 1 + 1 + 1 = 6. \end{aligned}$$

If we use the alphabet-weighted costs above, the optimal operation sequence is the same as before and the edit distance is

$$d(S_3, S_4) = 0.01 + 0.02 + 0.05 + 0.0125 + 0.10 + 0.0125 = 0.205.$$

$\square$

Consider a set of event types $\mathcal{E} = \{A, B, C, \ldots\}$ and $S_1 = \langle A, B, C, e_1, e_2, \ldots, e_m \rangle$ and $S_2 = \langle f_1, \ldots, f_m, A, B, C \rangle$ where each event e_i and f_j are in the set $\mathcal{E} \setminus \{A, B, C\}$ for $\forall i, j$, and no $e_i = f_j$. The distance $d(S_1, S_2)$ will be the larger the longer the sequences are. Because the occurrence times of the events do not influence the comparison of the sequences, the sequence S_1 would be said to be as similar with the sequence S_2 as, for example, a sequence $S_3 = \langle q_1, q_2, q_3, A, B, C, q_4, \ldots, q_m \rangle$. Even if all the sequences have a short common subsequence, in their entirety they are very different. Therefore, it seems obvious why also the occurrence times of the events should be considered when determining the similarity between event sequences. Still, in some cases finding short common subsequences with high similarity might be useful. In this thesis we will not discuss this problem any further; see [Gus97, SM97] for studies on this problem in molecular biology.

Variations

The set of edit operations for computing the distance between event type sequences can be extended to allow a *substitution* of an event with another, if that operation is considered to be natural. If the set $\mathcal{E}$ of event types has a metric $h_{\mathcal{E}}$ defined on it, one might define that the cost of transforming an event e to another event e' is $h_{\mathcal{E}}(e, e') + b$, where b is a constant. Such a metric $h_{\mathcal{E}}$ could, for example, be one of the attribute similarity measures defined in Chapter 3. Note that a substitution can be accomplished by a deletion followed by an insertion. If we want that substitutions of events are at least sometimes useful, the cost $h_{\mathcal{E}}(e, e') + b$ should be less than the sum of costs of a deletion and an insertion.

Assume then that we have two short event type sequences S_1 and S_2 that have no events of a common type at all, and two longer event type sequences S_3 and S_4 that differ only with few events. Intuitively, the two longer sequences are more similar to each other than the two short sequences. Because for transforming the event type sequence S_1 to the sequence S_2 we need only a few operations, the edit distance $d(S_1, S_2)$ can be less than the edit distance $d(S_3, S_4)$ which is against our intuition. If we want to avoid such a situation and eliminate the influence of the lengths of the event type sequences, we can normalize each edit distance $d(S, S')$ by a factor $\sum_{e_i \in S} \mathtt{c}(\mathtt{Del}(e_i)) + \sum_{f_j \in S'} \mathtt{c}(\mathtt{Ins}(f_j))$.

4.2.2 Event sequences

A more interesting problem than the similarity between event type sequences is to define the similarity between event sequences where the occurrence times of events are taken into account. Also in this case, the similarity between sequences is based on comparison of event types, but in addition to that the occurrence times of the events must be considered. This makes the definition of the event sequence distance more difficult.

Edit operations
In computing the edit distance between event sequences with occurrence times we have chosen to use a set $\mathcal{O}$ of three operations:

1. $\texttt{Ins}\,(e,t)$ that inserts an event of the type e to time t, i.e., adds an event (e,t) to the sequence.
2. $\texttt{Del}\,(e,t)$ that deletes an event of the type e from time t, i.e., deletes an event (e,t) from the sequence.
3. $\texttt{Move}\,(e,t,t^{'})$ that changes the occurrence time of the event (e,t) from time t to time $t^{'}$.

These operations were chosen because they are very natural. Intuition says that two event sequences $\mathcal{S}_1$ and $\mathcal{S}_2$ are similar if a lot of the events in them match and the optimal operation sequence $\hat{O}$ contains only few insertions and deletions and, of course, only short moves.

Example 4.12 Consider $\mathcal{E} = \{A, B, C, D, E\}$ and two event sequences

$$\mathcal{S}_1 = \langle (A,1),(B,3),(A,4)(C,5),(B,9),(D,11)\rangle$$

and

$$\mathcal{S}_2 = \langle (A,2),(B,5),(C,8),(C,12),(A,13),(D,15)\rangle$$

(also represented in Figure 4.3). An operation sequence

$$\begin{array}{l}\texttt{Move}\,(A,1,2),\ \texttt{Move}\,(B,3,5),\ \texttt{Del}\,(A,4),\\ \texttt{Move}\,(C,5,8),\ \texttt{Del}\,(B,9),\ \texttt{Ins}\,(C,12),\\ \texttt{Ins}\,(A,13),\ \texttt{Move}\,(D,11,15)\end{array}$$

is only one of the many possible operation sequences transforming the sequence $\mathcal{S}_1$ to the sequence $\mathcal{S}_2$. □

Costs of operations
There are several ways of defining the edit operation costs for event sequences with occurrence times. One possibility is to use unit costs for insertions and deletions of events, i.e., the cost of an insertion is $\texttt{c}\,(\texttt{Ins}\,(e,t)) = 1$ and, the cost of a deletion $\texttt{c}\,(\texttt{Del}\,(e,t)) = 1$. Because some types of events occur more often than the others, it might be more natural, if the costs were dependent on the event type. Similarly to the case of event type sequences, an alphabet-weighted cost of inserting an event into an event sequences with occurrence times can be defined by

$$\texttt{c}\,(\texttt{Ins}\,(e,t)) = w(e),$$

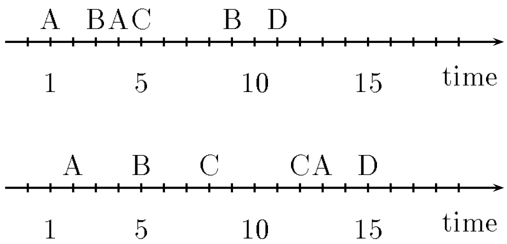

Figure 4.3: The event sequences $\mathcal{S}_1$ and $\mathcal{S}_2$ of Example 4.12 on the time axis.

where $w(e)$ is a constant value proportional to $occ(e)^{-1}$, where $occ(e)$ is the number of occurrences of an e-type event in a long reference sequence from the same application area. With this definition the cost of adding a rare event into the sequence is higher than the cost of adding a common event. The cost of deleting an event from an event sequence is defined to be the same as the cost of an Ins-operation, i.e., $\mathtt{c}\,(\mathtt{Del}\,(e,t)) = w(e)$.

A more difficult problem is defining the cost of a Move-operation. Both in the case of unit costs and in the case of alphabet-weighted costs of inserting or deleting an event, we define the cost of a Move-operation as

$$\mathtt{c}\,(\mathtt{Move}\,(e,t,t')) = V \cdot |\, t - t'\,|$$

where V is a constant and $|\,t - t'|$ is the length of the move. With this definition a short move has a lower cost than a long move. This definition assumes that the occurrence times of events have approximately the same magnitude in both compared sequences. Without this assumption, sequences such as $\langle(A,1),\ (B,2),\ (C,5)\rangle$ and $\langle(A,101),\ (B,102),\ (C,105)\rangle$ would be considered to be very far from each other.

Parameters

It is clear that the parameter values used in defining the costs influence the value of the distance measure. Therefore, it is important that the parameter values are chosen so that real similarity between sequences is caught properly.

In some applications and cases it may be useful, or even necessary, to limit the length of moves even more explicitly. This bound W can be a predefined value, *window size*, in given time units. It can also be some function of the occurrence times in the sequences compared, for example, the length of the longer sequence. With this bound, the cost of moving an event in time is always $\mathtt{c}\,(\mathtt{Move}\,(e,t,t')) \leq V \cdot W$.

The parameter V used in alpha-weighted costs has some logical restrictions. In the case of unit costs for inserting and deleting an event, we should have $V \leq 2$. If $V > 2$, moving an event is never useful: instead, one can always first delete and then insert an event. If we use alphabet-weighted costs, we should, for similar reasons, have $V \leq 2 \cdot w(e)$ for all the event types $e \in \mathcal{E}$. The highest value of V that satisfies this condition is $2 \cdot \mathsf{min\_w}$ where $\mathsf{min\_w} = min\{w(e)\}$. If $V = 2 \cdot \mathsf{min\_w}$, then for an event of type e which has the minimum weight $\mathsf{min\_w}$ the length of the longest useful move is one time unit. The length of the longest useful move for any other type of events is determined by equation

$$|t - t^{'}| \leq \frac{w(e)}{\mathsf{min\_w}}$$

where $w(e)$ is dependent on the event type e. The absolutely longest moves are useful for an event of a type e with the maximal value of $w(e)$ among all event types.

Similarly to event type sequences, if two event sequences $\mathcal{S}_1$ and $\mathcal{S}_2$ have no events of similar type, i.e., there are no events $e_i \in \mathcal{S}_1$ and $f_j \in \mathcal{S}_2$ such that $e_i = f_j$, the distance between the two event sequences is

$$\sum_{e_i \in \mathcal{S}_1} \mathtt{c}\,(\mathtt{Del}\,(e_i)) + \sum_{f_j \in \mathcal{S}_2} \mathtt{c}\,(\mathtt{Ins}\,(f_j)).$$

The other extreme distance value, $d(\mathcal{S}_1, \mathcal{S}_2) = 0$, is reached when the two sequences are identical. That is when all the events in the first sequence match in right order the events in the second sequence, i.e., $e_i = f_i$ and $t_i = u_i$ for all i. This also means that the sequences have to be equally long.

Example 4.13 Let $\mathcal{E} = \{A, B, C, D, E\}$ be a set of event types, $\mathcal{S}_1$ and $\mathcal{S}_2$ the two event sequences in Example 4.12 and the window size $W = 20$. If we use the constant value $V = 1/W = 0.05$ the cost of moving an event is $\mathtt{c}\,(\mathtt{Move}\,(e, t, t^{'})) = 0.05 \cdot |\,t - t^{'}|$. The cost of the maximum length move (20 time units) is $0.05 \cdot 20 = 1$. Using the unit costs for `Ins`- and `Del`-operations and the above cost for a `Move`-operation, the optimal operation sequence $\hat{O}$ is

$$\mathtt{c}\,(\mathtt{Move}\,(A,1,2)),\ \mathtt{c}\,(\mathtt{Move}\,(B,3,5)),\ \mathtt{c}\,(\mathtt{Del}\,(A,4)),\ \mathtt{c}\,(\mathtt{Move}\,(C,5,8))$$
$$\mathtt{c}\,(\mathtt{Del}\,(B,9)),\ \mathtt{c}\,(\mathtt{Ins}\,(C,12)),\ \mathtt{c}\,(\mathtt{Ins}\,(A,13)),\ \mathtt{c}\,(\mathtt{Move}\,(D,11,15))$$

and the edit distance between the two sequences is

$$\begin{aligned} d(\mathcal{S}_1, \mathcal{S}_2) &= \mathtt{c}\,(\mathtt{Move}\,(A,1,2)) + \mathtt{c}\,(\mathtt{Move}\,(B,3,5)) \\ &\quad + \mathtt{c}\,(\mathtt{Del}\,(A,4)) + \mathtt{c}\,(\mathtt{Move}\,(C,5,8)) \\ &\quad + \mathtt{c}\,(\mathtt{Del}\,(B,9)) + \mathtt{c}\,(\mathtt{Ins}\,(C,12)) \\ &\quad + \mathtt{c}\,(\mathtt{Ins}\,(A,13)) + \mathtt{c}\,(\mathtt{Move}\,(D,11,15)) \\ &= 0.05 \cdot 1 + 0.05 \cdot 2 + 1 + 0.05 \cdot 3 + 1 \\ &\quad + 1 + 1 + 0.05 \cdot 4 = 4.5. \end{aligned}$$

Assume that we have the following numbers of occurrences of event types and the alphabet-weighted costs for an insertion and a deletion of an event:

e	$occ(e)$	$\texttt{c}(\texttt{Ins}(e)) =$ $\texttt{c}(\texttt{Del}(e))$
A	100	0.0100
B	50	0.0200
C	20	0.0500
D	80	0.0125
E	10	0.1000

If we use a constant $V = (2 \cdot \texttt{min\_w})/W$, the cost of moving an event is $\texttt{c}(\texttt{Move}(e, t, t')) = [(2 \cdot 0.01)/20] \cdot |t - t'| = 0.001 \cdot |t - t'|$. The cost of a Move-operation with the maximal length 20 is, therefore, $0.001 \cdot 20 = 0.02$. The optimal edit operation sequence $\hat{O}$ is now as with the unit cost case above, and the edit distance between the two sequences is

$$\begin{aligned} d(\mathcal{S}_1, \mathcal{S}_2) &= 0.001 \cdot 1 + 0.001 \cdot 2 + 0.01 + 0.001 \cdot 3 \\ &\quad +0.02 + 0.05 + 0.01 + 0.001 \cdot 4 = 0.10. \end{aligned}$$

Another possibility is to use a constant $V = 2 \cdot \texttt{min\_w}$. The cost of moving an event is then $0.02 \cdot |t - t'|$. The cost of maximal length (20 time units) move is 0.4. This means that moving an event is not always cost-effective. The edit distance between the two sequences with these parameter values is

$$\begin{aligned} d(\mathcal{S}_1, \mathcal{S}_2) &= \texttt{c}(\texttt{Move}(A, 1, 2)) + \texttt{c}(\texttt{Move}(B, 3, 5)) \\ &\quad +\texttt{c}(\texttt{Del}(A, 4)) + \texttt{c}(\texttt{Move}(C, 5, 8)) \\ &\quad +\texttt{c}(\texttt{Del}(B, 9)) + \texttt{c}(\texttt{Ins}(C, 12)) \\ &\quad +\texttt{c}(\texttt{Ins}(A, 13)) + \texttt{c}(\texttt{Del}(D, 11)) \\ &\quad +\texttt{c}(\texttt{Ins}(D, 15)) \\ &\quad 0.02 \cdot 1 + 0.02 \cdot 2 + 0.01 + 0.02 \cdot 3 \\ &\quad +0.02 + 0.05 + 0.01 + 0.0125 + 0.0125 = 0.235. \end{aligned}$$

The optimal operation sequence $\hat{O}$ is different from the case with the constant value $V = (2 \cdot \texttt{min\_w})/W$. Moving an event D for 4 time units now costs 0.08 and deleting it from the first sequence and adding it to the other just 0.025, and therefore, deleting and inserting is preferred to moving the event.

Consider then two other event sequences $\mathcal{S}_3 = \langle (A, 6), (B, 8), (C, 12) \rangle$ and $\mathcal{S}_4 = \langle (D, 2), (E, 7), (D, 9) \rangle$. These two sequences have no events of a similar type, and the optimal operation sequence transforming the sequence $\mathcal{S}_3$ to the sequence $\mathcal{S}_4$ is

$$\begin{aligned} &\texttt{c}(\texttt{Del}(A)),\ \texttt{c}(\texttt{Del}(B)),\ \texttt{c}(\texttt{Del}(C)) \\ &\texttt{c}(\texttt{Ins}(D)),\ \texttt{c}(\texttt{Ins}(E)),\ \texttt{c}(\texttt{Ins}(D)) \end{aligned}$$

Therefore, the edit distance between them with unit operation costs for inserting and deleting an event and a cost $0.05 \cdot |t - t'|$ of moving an event in time is

$$
\begin{aligned}
d(\mathcal{S}_3, \mathcal{S}_4) &= \mathtt{c}(\mathtt{Del}(A)) + \mathtt{c}(\mathtt{Del}(B)) + \mathtt{c}(\mathtt{Del}(C)) \\
&\quad + \mathtt{c}(\mathtt{Ins}(D)) + \mathtt{c}(\mathtt{Ins}(E)) + \mathtt{c}(\mathtt{Ins}(D)) \\
&= 1 + 1 + 1 + 1 + 1 + 1 = 6
\end{aligned}
$$

which is the same result as with event type sequences (Example 4.11). Also the optimal operation sequence and the distance $d(\mathcal{S}_3, \mathcal{S}_4)$ with alphabet-weighted costs are the same as in the case of corresponding event type sequences. □

Consider a set of event types $\mathcal{E} = \{A, B, C, \ldots\}$ and two event sequences $\mathcal{S}_2 = \langle (e_1, t_1),\ \ldots,\ (e_m, t_m),\ (A, 57),\ (B, 58),\ (C, 60) \rangle$ and $\mathcal{S}_2 = \langle (A, 5),\ (B, 6),\ (C, 8),\ (f_1, t_1),\ \ldots,\ (f_m, t_m) \rangle$ where each e_i and f_j in the set $\mathcal{E} \setminus \{A, B, C\}$ for $\forall\ i, j$, and no $e_i = f_j$. The distance $d(S_1, S_2)$ will be rather large because the sequences have so many mismatching events. Also the moves would be very long: 52 time units. When taking into account the occurrence times of the events, the sequence $\mathcal{S}_1$ would be more similar to the sequence

$$
\begin{aligned}
\mathcal{S}_3 \;=\; & \langle (q_1, t_1),\ (q_2, t_2),\ (q_3, t_3),\ (q_4, t_4)\ (A, 13),\ (B, 14),\ (C, 16), \\
& (q_5, t_5),\ \ldots,\ (q_m, t_m), \rangle
\end{aligned}
$$

than to the sequence $\mathcal{S}_2$. This is the difference in determining the similarity between two event sequences to determining the similarity between two event type sequences. Still, there is a subsequence in all the three sequences where the relative time differences between events are the same. In this thesis we will not discuss the problem of finding such local similarities any further.

Variations

The model for defining edit operations and their costs for computing edit distance between event sequences can easily be extended to allow for an edit operation that changes the type of an event, i.e., a substitution of events. If the set $\mathcal{E}$ of event types has a metric $h_{\mathcal{E}}$ defined on it, one could, for example, define that the cost of transforming an event (e, t) to another event (e', t') is $h_{\mathcal{E}}(e, e') + b$, where b is a constant. As a metric $h_{\mathcal{E}}$ we could use, for example, one of the distance measures presented in Chapter 3. This definition, however, does not take into account the possible difference in times t and t'. One solution to this problem might be combining the costs of changing the event type and moving the event. If the substitutions of events are to be useful, at least sometimes, the cost of a substitution should be less than the sum of deletion and insertion costs.

Assume then that we have two short event sequences $\mathcal{S}_1$ and $\mathcal{S}_2$ that have no events of a common type at all, and two longer event sequences $\mathcal{S}_3$ and $\mathcal{S}_4$ that differ only with few events. Intuitively, the two longer sequences are more similar to each other than the two short sequences. Because for transforming the event sequence $\mathcal{S}_1$ to the sequence $\mathcal{S}_2$ we need only a few operations, the edit distance

$d(\mathcal{S}_1, \mathcal{S}_2)$ can be less than the edit distance $d(\mathcal{S}_3, \mathcal{S}_4)$. If we want to avoid such a situation and eliminate the influence of the lengths of the event sequences, we can normalize each edit distance $d(\mathcal{S}, \mathcal{S}')$ by a factor $\sum_{e_i \in \mathcal{S}} \mathtt{c}(\mathtt{Del}(e_i)) + \sum_{f_j \in \mathcal{S}'} \mathtt{c}(\mathtt{Ins}(f_j))$.

4.3 Algorithm for event sequence similarity

We use a fairly typical *dynamic programming method* [Aho90, CR94, Ste94, Gus97] for computing the weighted edit distance between two event sequences and finding the optimal operation sequence of transforming the first event sequence to the other. The dynamic programming approach has three essential components: a *recurrence relation*, a *tabular computation* and a *traceback* [Gus97].

Given the sequences $\mathcal{S}_1$ and $\mathcal{S}_2$, we use $\mathsf{r}(i,j)$ to denote the minimum cost of the operations needed to transform the first i events of the sequence $\mathcal{S}_1$ into the first j events of the sequence $\mathcal{S}_2$. The base conditions and the recurrence relation set a recursive relationship between the value $\mathsf{r}(i,j)$ for all positive i and j and the value r with pairs of indexes smaller than i and j. The weighted edit distance between the sequences $\mathcal{S}_1$ and $\mathcal{S}_2$ is, therefore, $\mathsf{r}(m,n)$, where m is the number of events in the sequence $\mathcal{S}_1$ and n the number of events in the sequence $\mathcal{S}_2$. We use the following base conditions and recurrence relation.

Definition 4.8 Let $\mathcal{S}_1 = \langle (e_1,t_1),(e_2,t_2),(e_3,t_3),\ldots,(e_m,t_m)\rangle$ and $\mathcal{S}_2 = \langle (f_1,u_1),(f_2,u_2),(f_3,u_3),\ldots,(f_n,u_n)\rangle$ be two event sequences, or similarly $S_1 = \langle e_1,e_2,e_3,\ldots,e_m\rangle$ and $S_2 = \langle f_1,f_2,f_3,\ldots,f_n\rangle$ two event type sequences. The base conditions and the recurrence relation for the value $\mathsf{r}(i,j)$ are

$$\begin{aligned}
\mathsf{r}(0,0) &= 0 \\
\mathsf{r}(i,0) &= \mathsf{r}(i-1,0) + w(e_i) \\
\mathsf{r}(0,j) &= \mathsf{r}(0,j-1) + w(f_j) \\
\mathsf{r}(i,j) &= min\{\, \mathsf{r}(i-1,j) + w(e_i), \\
&\quad\ \mathsf{r}(i,j-1) + w(f_j), \\
&\quad\ \mathsf{r}(i-1,j-1) + \mathsf{k}(i,j)\,\}
\end{aligned}$$

where $w(e_i)$ and $w(f_j)$ are costs of inserting (deleting) of a e_i-type or f_j-type event, respectively. In the case of event type sequences $\mathsf{k}(i,j)$ is defined as

$$\mathsf{k}(i,j) = \begin{cases} 0, & \text{if } e_i = f_j \\ w(e_i) + w(f_j), & \text{if } e_i \neq f_j, \end{cases}$$

For the case of event sequences with occurrence times $\mathsf{k}(i,j)$ is defined as

$$\mathsf{k}(i,j) = \begin{cases} V \cdot |t_i - u_j|, & \text{if } e_i = f_j \\ w(e_i) + w(f_j), & \text{if } e_i \neq f_j. \end{cases}$$

□

The second component of dynamic programming approach is using the basic conditions and the recurrence relation to efficiently compute the value $\mathsf{r}(m, n)$, i.e., the edit distance between sequences $\mathcal{S}_1$ and $\mathcal{S}_2$. We use a *bottom-up* approach to compute the distance. This means the values $\mathsf{r}(i, j)$ are saved in a $(m+1) \times (n+1)$ dynamic programming table and the values $\mathsf{r}(i, j)$ are computed for increasing values of indexes.

The dynamic programming table can be filled one column at time, in order of increasing i. In our approach, however, we have chosen to fill the table in one row at time, in order of increasing j. First, we set up the boundary values of the table. For each cell in the zeroth column the value $\mathsf{r}(i, 0)$ is the sum of the costs of deleting the first i events of the sequence $\mathcal{S}_1$. Similarly, each cell in the zeroth row of the table has the value $\mathsf{r}(0, j)$ which is the sum of the costs of inserting the first j events of the sequence $\mathcal{S}_2$. Later, when computing the value $\mathsf{r}(i, j)$, we know already the table values $\mathsf{r}(i-1, j)$, $\mathsf{r}(i, j-1)$ and $\mathsf{r}(i-1, j-1)$. The value of $\mathsf{r}(i, j)$ may be obtained by adding to $\mathsf{r}(i-1, j)$ the cost of deletion of an event e_i from the sequence $\mathcal{S}_1$, by adding to $\mathsf{r}(i, j-1)$ the cost of insertion of an event f_j to the sequence $\mathcal{S}_2$, or by adding to $\mathsf{r}(i-, j-1)$ the cost $\mathsf{k}(i, j)$ of transforming an event e_i in the sequence $\mathcal{S}_1$ to an event f_j in the sequence $\mathcal{S}_2$. The cost $\mathsf{k}(i, j)$ depends on whether $e_i = f_j$ or not. Because $\mathsf{r}(i, j)$ is the minimum cost of transforming the first i events of the sequence $\mathcal{S}_1$ to the first j events of the sequence $\mathcal{S}_2$, it is clear that we have to choose the cheapest of the results above as the value of $\mathsf{r}(i, j)$.

Example 4.14 Consider two event type sequences $S_1 = \langle A, B, A, C, B, D \rangle$ and $S_2 = \langle A, B, C, C, A, D \rangle$ in Example 4.8. Assume then that we have an operation set $\mathcal{O} = \{\texttt{Ins}, \texttt{Del}\}$ and the operations have unit costs.

The dynamic programming table r used in the computation of the edit distance between the sequences S_1 and S_2 is given in Figure 4.4. The value $\mathsf{r}(4, 4)$ in the table, for instance, is 2. Because the event e_4 in the sequence S_1 and the event f_4 in the sequence S_2 are both C, the value of $\mathsf{r}(4, 4)$ can be obtained either from $\mathsf{r}(3, 3)$ with $\mathsf{k}(4, 4) = 0$ or from $\mathsf{r}(4, 3)$ by inserting an event of type C. $\square$

The edit distance between two event sequences (or event type sequences) is computed by using Algorithm 4.1. The algorithm uses a dynamic programming table where the cells are filled according to the base conditions and the recurrence relation given in Definition 4.8.

r(i, j)	j	0	1	2	3	4	5	6
i			A	B	C	C	A	D
0		0	1	2	3	4	5	6
1	A	1	0	1	2	3	4	5
2	B	2	1	0	1	2	3	4
3	A	3	2	1	2	3	2	3
4	C	4	3	2	1	2	3	4
5	B	5	4	3	2	3	4	5
6	D	6	5	4	3	4	5	4

Figure 4.4: The dynamic programming table used to compute the edit distance between event type sequences $\langle A, B, A, C, B, D\rangle$ and $\langle A, B, C, C, A, D\rangle$.

Algorithm 4.1 Edit distance between event sequences

Input: Two event sequences $\mathcal{S}_1$ and $\mathcal{S}_2$, and costs $w(e)$ of deleting/inserting an e-type event.

Output: Edit distance $d(\mathcal{S}_1, \mathcal{S}_2)$ between the given sequences.

Method:

```
1.   r(0,0) = 0;
2.   for i = 1 to m do
3.       r(i,0) = r(i − 1,0) + w(e_i); od;
4.   for j = 1 to n do
5.       r(0,j) = r(0,j − 1) + w(f_j); od;
6.   for i = 1 to m do
7.       for j = 1 to n do
8.           r(i,j) = min { r(i − 1,j) + w(e_i),
                           r(i,j − 1) + w(f_j),
                           r(i − 1,j − 1) + k(i,j) };
9.       od;
10.  od;
11.  output r(m,n);
```

If we want to extract the optimal operation sequence of the computed edit distance, we can add a traceback method to the algorithm. This is the third element of the dynamic programming approach. The easiest way to do this is to establish *pointers* in the dynamic programming table as the table values are computed [Gus97]. When the value $\mathsf{r}(i,j)$ of the cell (i,j) is computed, we set a pointer from the cell (i,j) to the cell $(i,j-1)$ if $\mathsf{r}(i,j) = \mathsf{r}(i,j-1) + w(f_j)$, a pointer from the cell (i,j) to the cell $(i-1,j)$ if $\mathsf{r}(i,j) = \mathsf{r}(i-1,j-1)+w(e_i)$, and a pointer from the cell (i,j) to the cell $(i-1,j-1)$ if $\mathsf{r}(i,j) = \mathsf{r}(i-1,j-1)+\mathsf{k}(i,j)$. Each cell in the zeroth row has a pointer to the cell on its left, and each cell in

r(i, j)	j	0	1	2	3	4	5	6
i			A	B	C	C	A	D
0		0	$\leftarrow 1$	$\leftarrow 2$	$\leftarrow 3$	$\leftarrow 4$	$\leftarrow 5$	$\leftarrow 6$
1	A	$\uparrow 1$	$\nwarrow 0$	$\leftarrow 1$	$\leftarrow 2$	$\leftarrow 3$	$\nwarrow\leftarrow 4$	$\leftarrow 5$
2	B	$\uparrow 2$	$\uparrow 1$	$\nwarrow 0$	$\leftarrow 1$	$\leftarrow 2$	$\leftarrow 3$	$\leftarrow 4$
3	A	$\uparrow 3$	$\nwarrow\uparrow 2$	$\uparrow 1$	$\nwarrow\leftarrow\uparrow 2$	$\nwarrow\leftarrow\uparrow 3$	$\nwarrow 2$	$\leftarrow 3$
4	C	$\uparrow 4$	$\uparrow 3$	$\uparrow 2$	$\nwarrow 1$	$\nwarrow\leftarrow 2$	$\nwarrow\leftarrow 3$	$\nwarrow\leftarrow\uparrow 4$
5	B	$\uparrow 5$	$\uparrow 4$	$\nwarrow\uparrow 3$	$\uparrow 2$	$\nwarrow\leftarrow\uparrow 3$	$\nwarrow\leftarrow\uparrow 4$	$\nwarrow\leftarrow\uparrow 5$
6	D	$\uparrow 6$	$\uparrow 5$	$\uparrow 4$	$\uparrow 3$	$\nwarrow\leftarrow\uparrow 4$	$\nwarrow\leftarrow\uparrow 5$	$\nwarrow 4$

Figure 4.5: A dynamic programming table with pointers for extracting the optimal operation sequence included.

the zeroth column a pointer to the cell just above it. In all the other cells, it is possible (and common) that there are more than just one pointer. An example of a dynamic programming table with pointers is given in Figure 4.5.

The pointers in the dynamic programming table allow an easy recovery of the optimal operation sequence: simply follow any path of pointers from the cell (m, n) to the cell $(0, 0)$. Each horizontal pointer is interpreted as an insertion of an event f_j into $\mathcal{S}_2$, each vertical pointer as a deletion of an event e_i from $\mathcal{S}_1$, and each diagonal edge as a match $(e_i = f_j)$ or a mismatch $(e_i \neq f_j)$. If there is more than one pointer from a cell, then a path can follow any of them. Hence, a traceback path from the cell (m, n) to the cell $(0, 0)$ can start simply by following any pointer out of the cell (m, n) and then be extended by following any pointer out of any cell encountered. This means that we can have several different optimal traceback paths. The traceback of the optimal operation sequence can be done with Algorithm 4.2.

Algorithm 4.2 Extracting the optimal operation sequence

Input: A table r of the distance between event sequences $\mathcal{S}_1$ and $\mathcal{S}_2$.
Output: An optimal operation sequence of transforming the sequence $\mathcal{S}_1$ to the sequence $\mathcal{S}_2$.
Method:

```
i = m; j = n;
for (i > 0) and (j > 0) do
    if r(i, j) = r(i − 1, j − i) + k(i, j) do
        if e_i = f_j do
            push Move (e_i, t_i, u_j) into the sequence Ô
        od;
        else do
            push Del (e_i) and Ins (f_j) into the sequence Ô;
        od;
        i = i − 1; j = j − 1;
    od;
    else do
        if r(i, j) = r(i − 1, j) + w(e_i) do
            push Del(e_i) into the sequence Ô;
            i = i − 1;
        od;
        else do
            if r(i, j) = r(i, j − 1) + w(f_j) do
            push Ins(f_j) into the sequence Ô;
            j = j − 1;
        od;
    od;
od;
output the optimal operation sequence Ô;
```

Complexity considerations

The size of the dynamic programming table is $(m+1)\times(n+1)$ when we consider sequences $\mathcal{S}_1$ and $\mathcal{S}_2$ of lengths m and n, respectively. It takes a constant number of cell examinations, aritmetic operations and comparisons to fill in one cell of the table. Therefore, Algorithm 4.1 takes $O(mn)$ time and $O(mn)$ space to compute. If the sequences compared are fairly short, the quadratic behavior of Algorithm 4.1 is not a problem. However, if the sequences are typically very long, it should be better to compute at least similarity between event type sequences with more efficient algorithms than just dynamic programming; see [Ste94] for some of such algorithms.

On each iteration in Algorithm 4.2 either the index i, the index j, or both of them, is decremented. This means that the maximum number of iterations is $m + n$, and that the extraction of the optimal operation sequence is done in $O(m + n)$ time.

4.4 Experiments

In this section we present some results of experiments on similarity between event sequences. In Subsection 4.4.1 we describe the data sets used in the experiments. The results obtained are discussed in Subsection 4.4.2. All the experiments were run on a PC with 233 MHz Pentium processor and 64 MB main memory, under the Linux operating system. The sequences of events resided in flat text files.

4.4.1 Data sets

The experiments on similarity between event sequences were made with two data sets: a telecommunication network alarm data and a log of WWW page requests.

Telecommunication alarm data
In our telecommunication data set there were 287 different alarm types and totally 73 679 alarms. The data was collected during 50 days, i.e., a time period covering over 7 weeks. On three days there occurred no alarms at all, and one day there were a total of 10 277 alarms. The number of occurrences of alarm types diversified a lot: from one occurrence to 12186 occurrences. The mean number of occurrences of alarm types was 257.

We selected several interesting alarm types from the data. For each of them we extracted all the preceding subsequences so that the events in the subsequences occurred at most 60 seconds before the given type of an event. For each selected alarm type we experimented with two sets of subsequences: the set of event type sequences and the set of event sequences. In the following we present only results obtained with interesting alarm types $1400, 1660, 7125$ and 7272. For events of type 7125 there are 29 preceding subsequences. By change, the sets of the preceding subsequences for the other three alarm types consist all of 9 subsequences. For other alarm types, the sets of subsequences had very different sizes, depending on the number of occurrences of the alarm type considered.

WWW log data
The log of WWW page requests used in our experiments was collected during 12 days in University of Helsinki from the references to the pages located in the WWW server of the Department of Computer Science. In the raw data there were totally 139 134 references, of which 105 454 succeeded. When all the references to different pictures were filtered out, there still were 45 322 references. The number of referred WWW pages was 6023. The number of references per a page varied from one to 1848 references: the mean number of references was

7.5, and a total of 2563 pages were referred only once. The number of referring hosts was 7633, and the number of references made by hosts varied from one to 4656 references. A total of 4189 hosts made just one reference whereas the mean of made references per a host was 6.

We selected from the data set many interesting WWW pages. For each chosen page we extracted from the whole sequence all subsequences that preceded references to the given page so that the referring host was the same with every event in the subsequence as it was for the interesting page. The time window where the events of subsequences were supposed to occur was chosen to be 10 minutes, i.e., 600 seconds. For each selected page we made tests with two sets of subsequences: the set of event type sequences and the set of event sequences. In the following we present results only for three WWW pages: a project page in the yearly report of the Department of Computer Science at University of Helsinki (the CSUH project page), a research group page of the Helsinki Graduate School of Computer Science and Engineering (the HeCSE research page), and a page of the KDD related links on the pages of the Data Mining research group at the Department (the KDD link page). The set of subsequences preceding a reference to the CSUH project page contains 11 sequences, to the HeCSE research page 17 sequences, and to the KDD link page 15 sequences. For all the other selected pages the number of sequences varied more, depending on the total number of references made to the given page.

4.4.2 Results and discussion

In our experiments we computed edit distances between sequences in each test set with unit and alphabet-weighted costs. The alphabet-weighted costs of `Ins`- and `Del`-operations were obtained by using the number of the occurrences of different event types in the whole sequences considered: the number of occurrences of different alarm types in the whole telecommunication sequence and the number of references made to different WWW pages.

In the case of event sequences, the parameter V needed in computing the costs of `Move`-operations were chosen to have the value $\frac{1}{W}$ with unit operation costs and the value $\frac{2 \cdot \mathsf{min\_w}}{W}$ with alphabet-weighted costs. This means that moving an event was always preferred to first inserting and then deleting one. If we would have used the parameter value $V = 2 \cdot \mathsf{min\_w}$ with the alphabet-weighted costs, the distances would have become larger and even the optimal operation sequence could have been different. Comparison of the distances obtained with these two parameter values for quite a few sequences, however, showed that in many cases the distances seemed to have a positive linear correlation. Therefore, the exact evaluation of how different values of the parameter V influence edit distances was omitted from this study.

For the WWW page request data as well as for the telecommunication data,

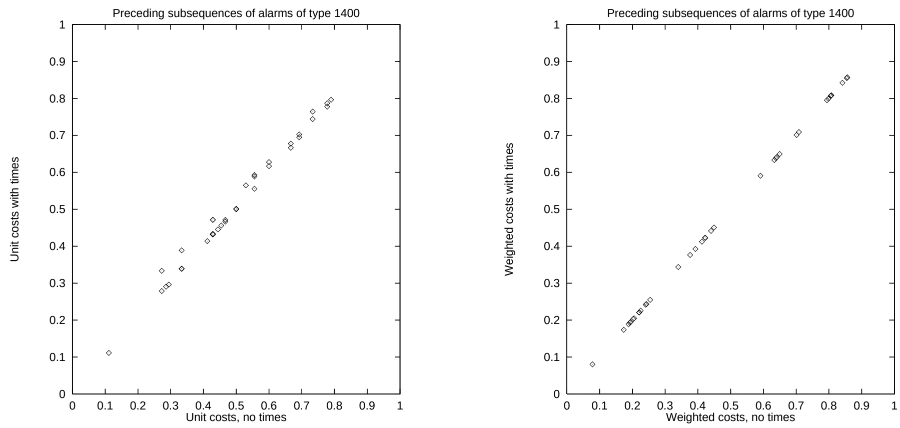

Figure 4.6: Comparison of distances between event type sequences and event sequences that precede events of type 1400 when the unit or alphabet-weighted operation costs are used.

we noticed that sequences within each test set considered were, typically, of very different lengths. Because we wanted to eliminate the influence of the lengths of sequences on the edit distances, we normalized them. As normalization factors we used the sum of the operation costs of first deleting all the events of the first sequence and then inserting all the events of the second sequence. That is, for each pair of event sequences $\mathcal{S}$ and $\mathcal{S}'$ we used a normalization factor $\sum_{e_i \in \mathcal{S}} \mathtt{c}(\mathtt{Del}(e_i)) + \sum_{f_j \in \mathcal{S}'} \mathtt{c}(\mathtt{Ins}(f_j))$.

As in the case of attributes, the actual edit distance values are often irrelevant. We can multiply or divide the distance values by any constant without modifying the properties of the measure. In many applications the only important thing is the relative order of the distance values. That is, as long as for all sequences $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ we have $d(\mathcal{S}_1, \mathcal{S}_2) < d(\mathcal{S}_1, \mathcal{S}_3)$ if and only if $d'(\mathcal{S}_1, \mathcal{S}_2) < d'(\mathcal{S}_1, \mathcal{S}_3)$, the measures d and d' behave in the same way.

Figure 4.6 shows how the distances between event type sequences and event sequences preceding an event of type 1400 are related to each other. The plot on the left describes the relation between distances with unit costs, and the plot on the right the relation between distances with alphabet-weighted costs. From both the plots we can see that the distances are positively linearly correlated. This is an expected behavior, as the costs of `Move`-operations are always small with the chosen parameter values. Similar phenomenon was observed also with other test sets.

We also compared how the distances between one kind of sequences behave with different operation costs. In Figure 4.7 there are four examples of such comparisons with event sequences; the results for event type sequences were very similar. The plot on the top left in Figure 4.7 describes how the distances of

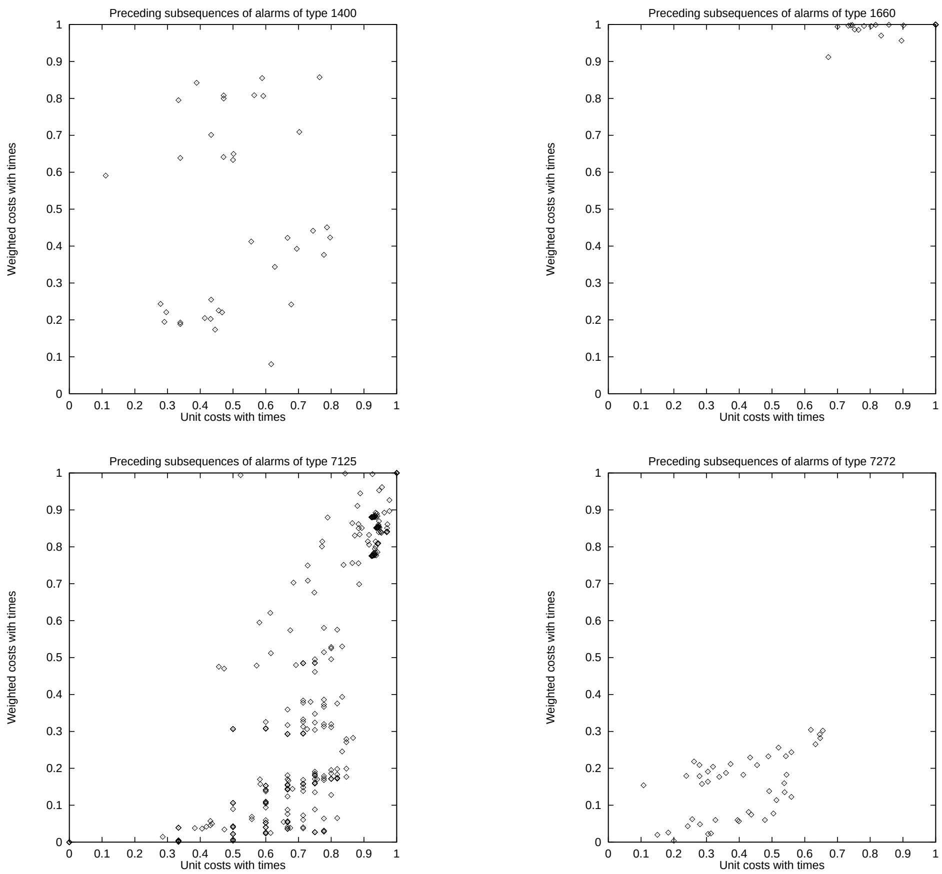

Figure 4.7: Comparison of distances with unit and alphabet-weighted operations costs for four sets of telecommunication alarm sequences with occurrence times.

event subsequences preceding alarms of type 1400 are related to each other. The distribution of points is wide, indicating that the costs used really influence on how similar two event sequences are considered to be. Two sequences that have a rather small distance with unit costs, can have a large distance with alphabet-weighted costs. This phenomenon also holds for the other direction: even if the distance between two sequences is small with alphabet-weighted costs, with unit costs the distance between them can be much larger. Consider, for example, three event sequences $\mathcal{S} = \langle(1553, 8), (691, 39), (690, 39), (690, 39), (1001, 39)\rangle$, $\mathcal{S}' = \langle(691, 39), (690, 39), (690, 39), (1001, 39)\rangle$ and $\mathcal{S}'' = \langle(690, 39), (1001, 39)\rangle$ preceding an alarm of type 1400. The distance between event sequences $\mathcal{S}$ and $\mathcal{S}'$ with unit costs is 0.1111, but with alphabet-weighted costs the distance between them is as high as 0.5909. On the other hand, the event sequences $\mathcal{S}'$ and $\mathcal{S}''$ have a distance 0.3333 with unit costs, but with alphabet-weighted costs a distance just 0.1882. This means that with different operation costs not only the distance values, but also the relative order of the pairwise edit distances

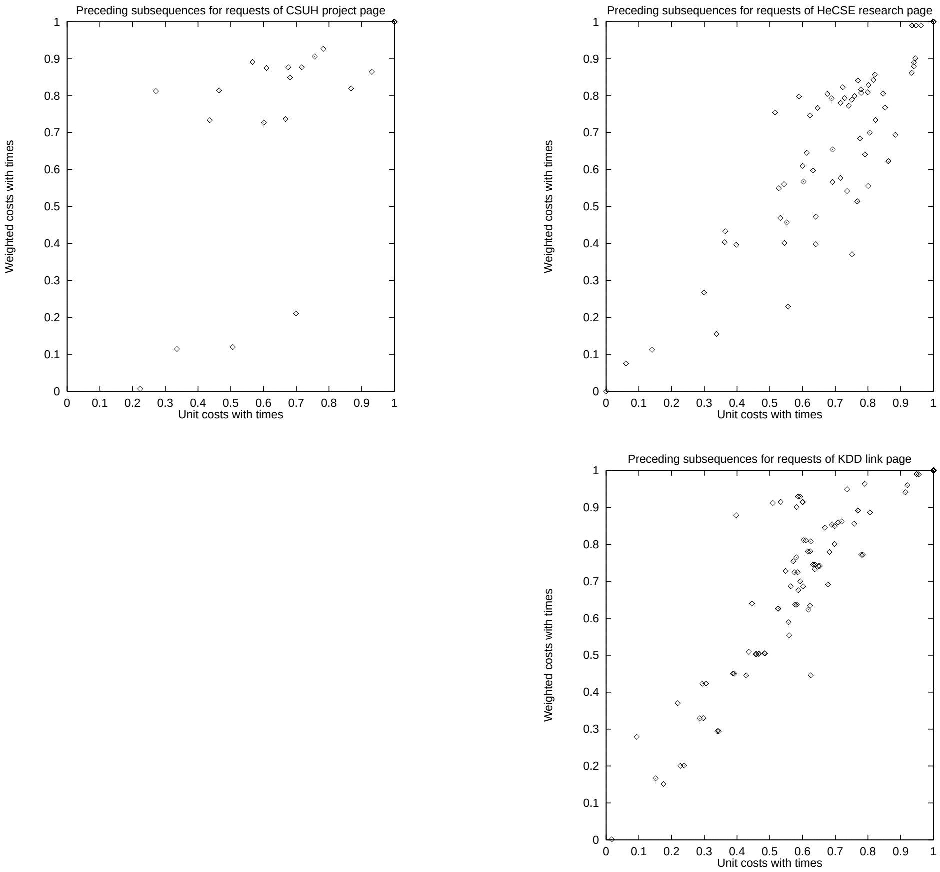

Figure 4.8: Comparison of distances with unit and alphabet-weighted operations costs for three sets of WWW page request sequences with occurrence times.

between sequences can be different.

The other three plots in Figure 4.7 show how the distances between event sequences preceding occurrences of the three other interesting alarm types are related to each other with different operation costs. Also in these cases, it is clear that using different operation costs can make the distances between two sequences very different. Similar conclusion can also be made from the plots in Figure 4.8 which show how the distances between event sequences in the case of the selected WWW pages behave when the distances are computed with different operation costs.

Earlier we found that the costs of `Move`-operations in each test case were small. This means that the main effect on the edit distances comes from events that cannot be moved because there is not an event of the same type in the other sequence. If these non-common events are very rare in the reference sequences, the alphabet weighted costs of inserting or deleting them are high, and the alphabet-weighted edit distance between the sequences is easily larger than the edit distance with unit operation costs. And if these non-common events occur very often in the reference sequence, the alphabet-weighted edit distance can become smaller than the edit distance with unit costs. This is the reason why the distance values differ so much with different operation costs, and therefore, the user must be able to choose such a set of operation costs that best suits the situation considered.

5 Clustering by similarity

In this chapter we consider methods for *clustering* of data, i.e., finding groups in data. In Section 5.1 we discuss the general properties of clustering, especially hierarchical clustering. Measures needed in the clustering process and evalution of the results are given in Section 5.2. The clustering algorithm used in the experiments is represented in Section 5.3. Sections 5.4.1 and 5.4.2 describe how attributes and event sequences, respectively, can be clustered by similarity. Some experimental results on the Reuters-21578 data, the student enrollment data, the telecommunication network alarm data and the WWW page log data are presented in Section 5.4.

5.1 Hierarchical clustering

Discovering structure and relationships within data is an important problem in many application areas. For example, in analysing market basket data it is interesting to find market segments, i.e., groups of customers with similar needs, or to find groups of similar products. On the other hand, in analyzing the behavior of a telecommunication network it could be useful to find typical situations preceeding severe failures in the network. Other examples of areas where clustering of objects has been found important are medicine (clustering of patients and diseases), biology (grouping of plants and animals), geography (clustering of regions), and chemistry (classification of compounds).

Definition 5.1 Let $\mathbf{O} = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$ be a set of objects and $\mathbf{C}$ a set of clusterings $\mathcal{C}$. A *clustering* $\mathcal{C}$ of $\mathbf{O}$ is a partition $\{c_1, c_2, \ldots, c_k\}$ where each *cluster* c_i is a subset of $\mathbf{O}$ so that $\bigcup_{i=1}^{k} c_i = \mathbf{O}$ and $c_i \cap c_j = \emptyset$ for $i \neq j$. The size of the clustering $\mathcal{C}$, i.e., the number of clusters in the clustering, is denoted as $|\mathcal{C}|$. A cluster c_i is called a *singleton cluster*, if it contains only one object, i.e., the size of the cluster is $|c_i| = 1$. □

In this chapter we use in examples mainly artifical data but give also some examples on biological data and market basket data.

Example 5.1 Let $\mathbf{O}$ be the set of animals. One possible clustering of $\mathbf{O}$ is to divide the animals in the groups of birds, fish, insects, mammals, and reptiles. Inside each of these clusters there can be several smaller clusters. For example, the cluster of reptiles contains smaller clusters such as snakes, lizards and crocodiles. □

Example 5.2 In a market basket data the object set $\mathbf{O}$ consists of the products sold in the supermarket. Possible clusters of the products are beverages, bread, conserves, dairy products, fish, frozen foods, fruits, meat, and vegetables. Of these, for example, the group of beverages can be divided to clusters like teas, coffees, beers and soft drinks. □

Research in the field of clustering has been extensive, and many different methods for grouping of data have been developed; see [And73, JD88, KR90] for overviews of cluster analysis. Each clustering method describes the structure of the data from one point of view. This means that different methods may produce different kinds of clusterings. A clustering produced by one method may be satisfactory for one part of the data, another method for some other part. Therefore, in many cases it may be useful to try several clustering methods on the data. One should also remember that the data may contain just one big cluster, or no clusters at all.

According to Definition 5.1 a clustering is a partition where each cluster contains at least one object and each object belongs to exactly one cluster, i.e., clusters are disjoint. A group of clustering techniques that find such clusterings are the *hierarchical clustering* methods. For example, in biology the hierachical clustering methods are widely used method for the classification of animals and plants. Instead of one single partition of the given objects, the hierarchical methods construct a sequence $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_{n-1}$ of clusterings. This sequence of clusterings is often represented as a *clustering tree*, also called a *dendrogram* [DJ76, DH73]. In a clustering tree leaves represent the individual objects and internal nodes the clusters. An example of a clustering tree is given in Figure 5.1.

There are two kinds of hierarchical clustering techniques: *agglomerative* and *divisive*. Difference between these techniques is the direction in which they construct the clusterings. An agglomerative hierarchical clustering algorithm starts from the situation where each object forms a cluster, i.e., we have n disjoint

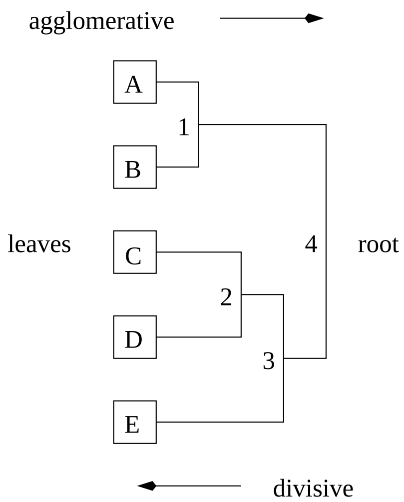

Figure 5.1: An example of a clustering tree.

clusters. Then in each step the algorithm merges the two most similar clusters until there is only one cluster left. A divisive hierarchical clustering algorithm, on the other hand, starts with one big cluster containing all the objects. In each step the divisive algorithm divides the most distinctive cluster into two smaller clusters and proceeds until there are n clusters each of which contains just one object. In Figure 5.1 the clustering trees produced by agglomerative and divisive methods are the same, but usually they are different [KR90]. In literature, the agglomerative methods are sometimes referred as *bottom-up* and the divisive methods as *top-down* hierarchical clustering. In the following sections we consider only agglomerative hierarchical clustering.

Because the hierarchical methods are conceptually simple and their theoretical properties are well understood, they are among the most popular clustering methods. One reason for their popularity lies in their way of treating the merges, or the splits, of clusters. Namely, once two clusters are merged by any agglomerative hierarchical algorithm, they are joint permanently. Similarly, when a cluster is split by a divisive algorithm, the two smaller clusters are separated permanently. This means that the number of different alternatives needed to examine in each clustering phase is reduced, and a computation time of the method is rather small. This property of keeping the merging and splitting decisions as permanent is, unfortunately, at the same time also the main disadvantage of hierarchical clustering methods: if the algorithm makes an erroneous decision on merging, or splitting, is impossible to correct it later.

5.2 Clustering measures

One of the main goals of every clustering method is to find, for the given set of objects, a set of clusters so that objects within each cluster are similar. This means that we want the clusters to be *tight*. Another main goal of clustering methods is to find a clustering where objects in different clusters are very dissimilar to each other, i.e., we want clusters to have a large *distance*. These two properties can be used to describe how good a clustering is.

Evaluation of the *quality* [FL85], or *goodness* [PR87, PR88] of a clustering can, of course, be done completely by a human analyst. However, to make this evaluation task easier, we can define numerical measures for each of the properties above. In the following sections we describe one possibility of defining the three clustering measures: distance, tightness and quality of a clustering. This approach is based on the clustering schema of [PR87].

5.2.1 Distance of clustering

Perhaps the most important of the clustering measures is the distance of a clustering. This measure describes how close, or far from each other the individual clusters in a clustering are. Therefore, we start by defining the distance between two clusters.

Definition 5.2 Let c_i and c_j be two clusters in a clustering $\mathcal{C}$. An *inter-cluster distance* $d(c_i, c_j)$ between two singleton clusters $c_i = \{\gamma_i\}$ and $c_j = \{\gamma_j\}$ is defined as a distance between objects γ_i and γ_j, i.e.,

$$d(c_i, c_j) = d_{\mathsf{f}}(\gamma_i, \gamma_j)$$

where d_{f} is a distance measure defined for the particular type of objects γ_i and γ_j.

Consider then a situation where at least one of the clusters c_i and c_j consists of two or more objects. Now the inter-cluster distance between the clusters c_i and c_j is a function F of the pairwise distances between objects when one of them is in the cluster c_i and the other in the cluster c_j, i.e.,

$$d(c_i, c_j) = F(\{d_{\mathsf{f}}(\gamma_k, \gamma_l) \,|\, \gamma_k \in c_i \text{ and } \gamma_l \in c_i\}).$$

This definition can also be applied to singleton clusters: there is just one pair of objects to compare. □

The choice of the distance function d_{f} depends on the type of the objects considered. For example, if the objects are attributes in a relation, any distance measure in Chapter 3 can be used as d_{f}. On the other hand, if the objects are event sequences (or event type sequences), the edit distance between event sequences (event type sequences) defined in Chapter 4 could be used as the distance measure d_{f} between objects.

The function F defining the inter-cluster distance between non-singleton clusters can be chosen in different ways. In this thesis we consider functions F corresponding to three common agglomerative hierarchical clustering methods: *single linkage*, *complete linkage*, and *average linkage* method. In all these methods, the inter-cluster distance between non-singleton clusters is defined differently.

The single linkage method, also referred as the *nearest-neighbor* method, is the oldest and simplest of the agglomerative clustering methods. The inter-cluster distance in this method is defined as the distance between the closest members of the two clusters, i.e.,

$$d(c_i, c_j) = min\,(\{d_{\mathsf{f}}(\gamma_k, \gamma_l) \,|\, \gamma_k \in c_i \text{ and } \gamma_l \in c_i\}).$$

The method is called single linkage because in each clustering phase two clusters are merged by the single shortest link between them (Figure 5.2a). Using this method we get clusters where every object in a cluster is more similar to all the

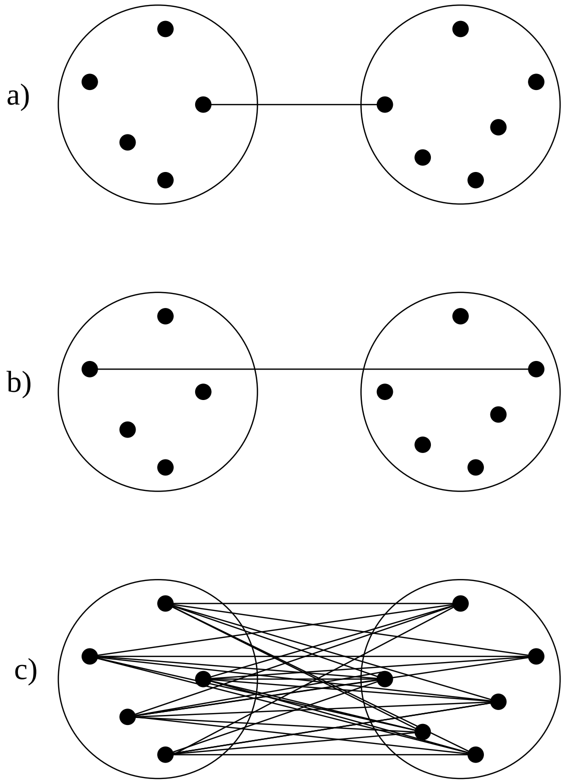

Figure 5.2: Inter-cluster distances with different agglomerative methods: a) single linkage, b) complete linkage, and c) group average linkage method.

other objects in the same cluster than to any other object not in that cluster. The problem with the single linkage method is its tendency to form elongated, serpentine like clusters (Figure 5.3a). This tendency is called a *chaining effect* [And73, DH73, KR90], and it can easily lead to a situation where two objects at opposite ends of the same cluster are extremely dissimilar. Of course, if the clusters really are elongated, this property of the single linkage method causes no problems.

An opposite method to the single linkage is a complete linkage clustering, also called the *furthest neighbor* clustering. In this method the inter-cluster distance between two clusters is defined as the distance between their farthest members (Figure 5.2b), i.e.,

$$d(c_i, c_j) = max\,(\{d_{\mathsf{f}}(\gamma_k, \gamma_l), |\, \gamma_k \in c_i \text{ and } \gamma_l \in c_i\,\}).$$

Now all the objects in a cluster are linked to each other at some maximum distance [And73], i.e., the longest distance needed to connect any object in one cluster to any object in the other cluster. This method tends to form compact clusters, but not necessarily well separated clusters (Figure 5.3b). In

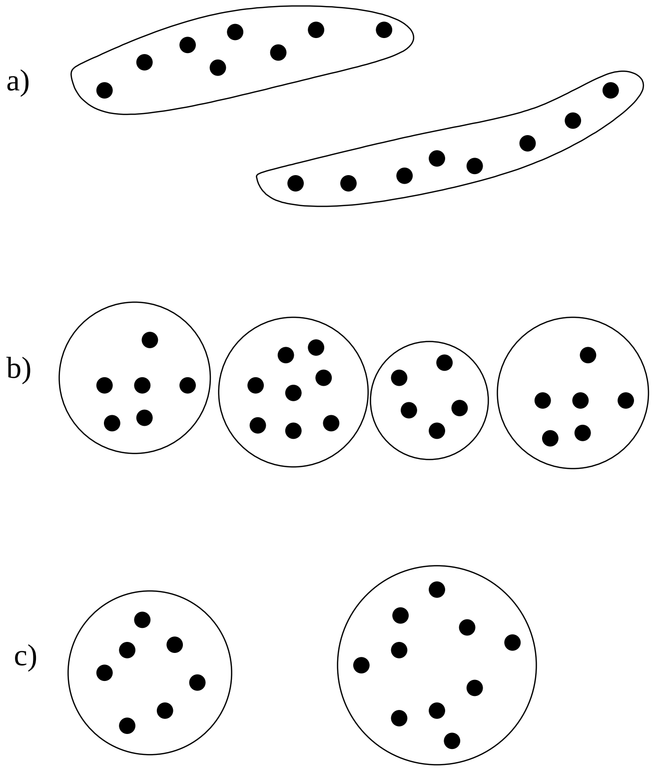

Figure 5.3: Some cluster types: a) elongated, b) compact, but not well separated, and c) ball-shaped clusters [KR90].

this method, forming of elongated clusters is highly discouraged, and if the real clusters are elongated, the resulting clusters can be meaningless [DH73].

The third agglomerative clustering method we consider is the average linkage method. In this method the inter-cluster distance of the clusters c_i and c_j is defined as

$$d(c_i, c_j) = avg\,(\{d_{\mathsf{f}}(\gamma_k, \gamma_l) \mid \gamma_k \in c_i \text{ and } \gamma_l \in c_i\}),$$

i.e., the distance is the mean of the pairwise distances between the objects in two clusters (Figure 5.2c). This method is aimed at finding roughly ball-shaped clusters (Figure 5.3c). In literature, this method is also referred as *average linkage between merged clusters* [And73], and the *unweighted pair-group average* method [DE83].

Example 5.3 Let $\mathbf{O} = \{A, B, C, D, E, F\}$ be a set of objects and the table

Object	A	B	C	D	E	F
A	0.00	0.10	0.45	0.50	0.80	0.70
B	0.10	0.00	0.40	0.50	0.90	0.10
C	0.45	0.40	0.00	0.10	0.50	0.20
D	0.50	0.50	0.10	0.00	0.30	0.40
E	0.80	0.90	0.50	0.30	0.00	0.15
F	0.70	0.10	0.20	0.40	0.15	0.00

present the pairwise distances between them according to some distance measure.

Assume then that in a clustering $\mathcal{C}$ we have two clusters: $c_1 = \{A, B, C, D\}$ and $c_2 = \{E, F\}$ (note that in reality this clustering is not necessarily produced by any of the three agglomerative methods). The inter-cluster distance between these clusters using the single linkage method is $d(c_1, c_2) = 0.10$, using the complete linkage method $d(c_1, c_2) = 0.90$, and the average linkage method $d(c_1, c_2) = 0.4875$. If the distance between any two objects is at most 1, then we can say that the single linkage method would define the cluster c_1 and c_2 very similar, the complete linkage method very dissimilar, and the average linkage method neither particular similar nor very dissimilar. □

The distance of a clustering is dependent on the inter-cluster distances of pairs of clusters, and it is defined as follows.

Definition 5.3 Let $\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$ be a clustering, c_i and c_j two clusters in the clustering $\mathcal{C}$, and $d(c_i, c_j)$ the distance between clusters c_i and c_j. The *distance of a clustering* $\mathcal{C}$ is a function $\mathcal{D} : \mathbf{C} \to \mathrm{IR}^+$, and it is defined as

$$\mathcal{D}(\mathcal{C}) = min_{1 \leq i,j \leq k, i \neq j}\{\, d(c_i, c_j)\,\},$$

i.e, the minimum distance value over all pairs of clusters. The distance of clustering is ∞, when $|\mathcal{C}| = 1$. □

The distance $\mathcal{D}$ of a clustering has a high value if all the clusters of it are very far from each other, and a low value, if at least two of the clusters are close to each other. Note that the distance $\mathcal{D}$ is defined as a function of clusterings, not of pairs of clusters.

Example 5.4 Consider the object set and the distance matrix of Example 5.3. Assume now that we have a clustering $\mathcal{C}' = \{c_1, c_2, c_3\}$ where $c_1 = \{A, B\}$ $c_2 = \{C, D\}$ and $c_3 = \{E, F\}$ (note that in reality this clustering is not necessarily produced by any of the three agglomerative methods). The inter-cluster distance between different clusters according to the three linkage methods are

Method	$d(c_1, c_2)$	$d(c_1, c_3)$	$d(c_2, c_3)$
single linkage	0.40	0.10	0.20
complete linkage	0.50	0.90	0.50
average linkage	0.46	0.63	0.35

Using the single linkage method, the distance of the clustering $\mathcal{C}'$ is, therefore, $\mathcal{D}(\mathcal{C}') = 0.10$. With the complete and the average linkage methods we get the distance values $\mathcal{D}(\mathcal{C}') = 0.50$ and $\mathcal{D}(\mathcal{C}') = 0.35$, respectively. The clustering $\mathcal{C}'$ would, therefore, be considered a bit better using the complete linkage method than using the average linkage method and much better than using the single linkage method. □

5.2.2 Tightness of clustering

Another important clustering measure is the tightness of a clustering. The value of this measure depends on how tight the individual clusters in the clustering are, i.e., how similar the objects in the clusters are. Therefore, we start by defining a measure for describing the tightness of a cluster.

Definition 5.4 Let $\mathbf{O}$ be a set of objects. The maximum distance between objects in $\mathbf{O}$ is denoted as $max\_dist = max\,\{\, d(\gamma_i, \gamma_j) \,|\, \gamma_i, \gamma_j \in \mathbf{O}\,\}$. The *tightness of a cluster* c_i is then defined as

$$tight(c_i) = \begin{cases} max\_dist, & \text{if } |c_i| = 1 \\ max\_dist - max\,\{\, d(\gamma_i, \gamma_j) \,|\, \gamma_i, \gamma_j \in c_i\,\}, & \text{if } |c_i| > 1 \end{cases}$$

□

According to this definition, the tightness of a cluster is high, if the distances between the objects belonging to that cluster are small. If there is, on the other hand, at least one pair of objects in a cluster that have a large distance, the tightness of the cluster is low.

Example 5.5 Consider the object set and the distance matrix of Example 5.3. The maximum distance between two objects in $\mathbf{O}$ is $max\_dist = 0.90$ which is also the tightness of every singleton cluster of the object set.

Assume then that we have a cluster $c = \{A, B\}$. The tightness of this cluster is $tight(c) = 0.90 - 0.10 = 0.80$. For another cluster $c' = \{B, C\}$ we get the tightness $tight(c') = 0.50$, and for a third cluster $c'' = \{A, E\}$ the tightness $tight(c'') = 0.10$. The higher the value $tight$ is the tighter the cluster is. Therefore, the cluster c is tighter than the clusters c' and c''. □

Now we can define the tightness of a clustering.

Definition 5.5 Let $\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$ be a clustering. The *tightness of a clustering* $\mathcal{C}$ is a function $\mathcal{T} : \mathbf{C} \rightarrow \mathbb{R}^+$ and is defined as

$$\mathcal{T}(\mathcal{C}) = min_{1 \leq i \leq k} \{ \, tight(c_i) \, \},$$

where k is the number of the clusters in the clustering. When $|\mathcal{C}| = 1$, the tightness of clustering is defined as zero, i.e., $\mathcal{T}(\mathcal{C}) = 0$. □

The tightness $\mathcal{T}$ of a clustering has a high value, when at least one of the clusters is very tight, i.e., it contains very similar objects. If all the clusters are tight, $\mathcal{T}$ has a low value. Note that the tightness $\mathcal{T}$ of a clustering is a function of clusterings, not of pairs of clusters.

Example 5.6 Consider the object set and the distance matrix of Example 5.3. For the clustering $\mathcal{C} = \{c_1, c_2\}$ in the same example, we have $\mathcal{T}(\mathcal{C}) = min\,\{0.4, 0.75\} = 0.4$. For the clustering $\mathcal{C}' = \{c_1, c_2, c_3\}$ in Example 5.4 the tightness of the clustering is $\mathcal{T}(\mathcal{C}') = min\,\{0.8, 0.8, 0.75\} = 0.75$. Hence, the clustering $\mathcal{C}'$ is tighter than the clustering $\mathcal{C}$. □

5.2.3 Quality of clustering

The quality of a clustering is a measure that describes how well both tightness and distance of a clustering have been achieved. We define the quality of a clustering as follows.

Definition 5.6 Let $\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$ be a clustering. The *quality of a clustering* $\mathcal{C}$ is a function $\mathcal{Q} : \mathbf{C} \rightarrow \mathbb{R}^+$, and given the tightness $\mathcal{T}$ and the distance $\mathcal{D}$ of a clustering, it can be defined as

$$\mathcal{Q}(\mathcal{C}) = min\,\{ \, \mathcal{T}(\mathcal{C}), \mathcal{D}(\mathcal{C}) \, \},$$

i.e., the minimum of the tightness and the distance of the clustering. □

In clustering of objects we want to find the best clustering according to the given quality function. In the case of Definition 5.6 this means searching for the clustering that maximizes the value of $\mathcal{Q}$, i.e., the minimum of $\mathcal{T}$ and $\mathcal{D}$. In Definition 5.6 both $\mathcal{T}$ and $\mathcal{D}$ are considered equally important. If we want to weight one of the two components more, we can use any other monotonically increasing function of the tightness $\mathcal{T}$ and the distance $\mathcal{D}$ than the minimum of them [PR87, PR88].

Example 5.7 Let the object set and the distance matrix be as in Example 5.3. Consider first the clustering $\mathcal{C} = \{c_1, c_2\}$ in the same example. Because there are only two clusters in $\mathcal{C}$, the value of $\mathcal{D}$ with each agglomerative method is the same as the inter-cluster distance between these two clusters, i.e.,

Method	$\mathcal{D}(\mathcal{C})$
single linkage	0.10
complete linkage	0.90
average linkage	0.49

The tightness of $\mathcal{C}$, on the other hand, is $\mathcal{T}(\mathcal{C}) = min\,\{0.40, 0.75\} = 0.40$. With the tightness value $\mathcal{T}(\mathcal{C})$ and the distance values $\mathcal{D}(\mathcal{C})$ above, the quality values of $\mathcal{C}$ with the different agglomerative methods are

Method	$\mathcal{Q}(\mathcal{C})$
single linkage	$min\,\{0.10, 0.40\} = 0.10$
complete linkage	$min\,\{0.90, 0.40\} = 0.40$
average linkage	$min\,\{0.49, 0.40\} = 0.40$

This means that the quality of $\mathcal{C}$ is considered to be better when using the complete or the average linkage method than the single linkage method.

Consider then the clustering $\mathcal{C}'$ in Example 5.4. The quality of $\mathcal{C}'$ with different methods is

Method	$\mathcal{Q}(\mathcal{C}')$
single linkage	$min\,\{0.10, 0.75\} = 0.10$
complete linkage	$min\,\{0.50, 0.75\} = 0.50$
average linkage	$min\,\{0.35, 0.75\} = 0.35$

Therefore, the quality of $\mathcal{C}'$ is in the case of the single linkage method is very low, and a bit better in the case of the average linkage method. The best quality value for the clustering is given by the complete linkage method. From the table above, we can also see that with every method the distance $\mathcal{D}$ of the clustering $\mathcal{C}'$ gives the value for the quality $\mathcal{Q}(\mathcal{C}')$. This just happened by change; generally, it is impossible to say which of the values $\mathcal{T}$ or $\mathcal{D}$ gives the value of $\mathcal{Q}$. □

5.3 Algorithm for hierarchical clustering

In this section we represent a clustering algorithm which is based on a common algorithm for agglomerative hierarchical clustering. In addition to forming the clusters, the given algorithm computes for each clustering the values of the clustering measures defined in Section 5.2. It also indicates which of the n clusterings produced is the clustering with the best quality value.

Algorithm 5.1 Agglomerative hierarchical clustering

Input: A set $\mathbf{O}$ of n objects and a matrix of pairwise distances between the objects.
Output: Clusterings $\mathcal{C}_0, \ldots, \mathcal{C}_{n-1}$ of the input set $\mathbf{O}$ where the clustering with the best quality is indicated.
Method:

$\mathcal{C}_0 =$ the trivial clustering of the input set $\mathbf{O}$;
compute the values of $\mathcal{T}$, $\mathcal{D}$ and $\mathcal{Q}$ for $\mathcal{C}_0$;
$\mathcal{C}_{best} = \mathcal{C}_0$;
$\mathcal{Q}(\mathcal{C}_{best}) = \mathcal{Q}(\mathcal{C}_0)$;
for $k = 1$ to $|\mathbf{O}| - 1$ **do**
    find $c_i, c_j \in \mathcal{C}_{k-1}$ so that the distance $d(c_i, c_j)$ is smallest;
    $\mathcal{C}_k = (\mathcal{C}_{k-1} - c_i - c_j) \cup merge(c_i, c_j)$;
    compute the distance $d(c_i, c_j)\ \forall\ c_i, c_j \in \mathcal{C}_k$;
    compute the values of $\mathcal{T}$, $\mathcal{D}$ and $\mathcal{Q}$ for $\mathcal{C}_k$;
    if $\mathcal{Q}(\mathcal{C}_k) > \mathcal{Q}(\mathcal{C}_{best})$ **then**
        $\mathcal{C}_{best} = \mathcal{C}_k$;
        $\mathcal{Q}(\mathcal{C}_{best}) = \mathcal{Q}(\mathcal{C}_k)$;
    end;
od;
output $\mathcal{C}_0, \ldots, \mathcal{C}_{n-1}$ and indicate $\mathcal{C}_{best}$ among them;

Algorithm 5.1 gets as input a finite set $\mathbf{O}$ of n objects and a matrix of pairwise distances between the objects. This means that executing the clustering algorithm is completely independent of how the distances between the objects were computed. The algorithm produces n different clusterings of the objects by starting with the trivial clustering $\mathcal{C}_0$ with singleton clusters.

On each iteration phase the algorithm searches those two clusters c_i and c_j that have the minimum distance in $\mathcal{C}_{k-1}$ and merges them. A new clustering $\mathcal{C}_k$ is formed by removing the two clusters and adding the new merged cluster, i.e., $\mathcal{C}_k$ is $\mathcal{C}_{k-1}$ with clusters c_i and c_j merged. Merging of clusters is continued until there is only one cluster left. After every merge the algorithm evaluates the tightness, the distance and the quality of the new clustering. The output of the algorithm is the sequence of clusterings $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_{n-1}$ among which the clustering with the best quality value is indicated.

Example 5.8 Consider the object set and the distance matrix of Example 5.3. Using Algorithm 5.1 and the three agglomerative methods represented in Section 5.2 we get the three clustering trees in Figure 5.4. All the trees are different. Difference between the clustering trees produced by complete and average linkage clustering is not big, only the last two clustering phases differ. The clustering tree produced by the single linkage method, however, is very different from the clustering trees produced by the other two methods.

The best clustering of the single linkage method contains two clusters: a cluster of objects $\{A, B, E, F\}$ and a cluster of objects $\{C, D\}$. The best clustering in the case of the complete linkage method has a size three. The clusters

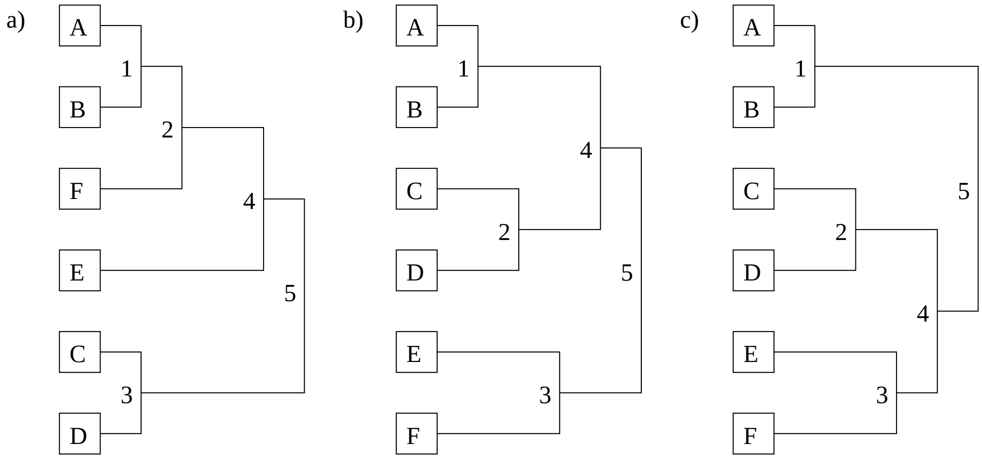

Figure 5.4: Clustering trees of an object set $\{A, B, C, D, E, F\}$ with three different clustering methods: a) single linkage, b) complete linkage, and c) average linkage method.

in this clustering are $c_1 = \{A, B\}$, $c_2 = \{C, D\}$ and $c_3 = \{E, F\}$. The group average method gives as the best clustering the clusters $\{A, B\}$ and $\{C, D, E, F\}$. Thus, the best clusterings indicated by the different methods are very different from each other. $\square$

The example 5.8 shows that the clustering algorithm can produce different clusterings with the three agglomerative methods. Also the best clusterings chosen can be very different, even if the structures of the clustering trees are similar.

According to [PR87], a clustering with the best quality can be found with an algorithm like Algorithm 5.1, if the clustering measures used fulfil the following three properties:

1. The tightness of a clustering is a monotone nonincreasing function under generalization, i.e., if one clustering is a generalization of another, then the more general clustering is at most as tight as the less general.

2. a) The distance of a clustering is the minimum of the pairwise inter-cluster distances and b) the inter-cluster distance function is a monotone function, i.e., if objects are added to clusters, the distance between these clusters cannot increase.

3. The quality of a clustering considers both the tightness and the distance of a clustering equally important and maximizes the minimum of them.

Our tightness function $\mathcal{T}$ and quality function $\mathcal{Q}$ fulfil the given criteria. Also the distance function $\mathcal{D}$ fulfils the property 2a). The property 2b) says that as the clusters grow, the distance between them should get smaller. Unfortunately, of the three agglomerative linkage methods presented, only the single linkage method fulfils this condition. This means that in the case of the complete and the average linkage methods we cannot be totally sure that our algorithm finds and indicates the best possible clustering.

Complexity considerations

The time and space complexity of Algorithm 5.1 can be estimated as follows. The size of the distance matrix is $n \times n$ rows. This means that in each phase of the algorithm searching of the closest pair of clusters takes $O(n^2)$ time. Because there are totally n clustering phases, the time complexity of the whole algorithm is $O(n^3)$. Because the distance matrix and the clusterings, expecially the best clustering, must be kept in the memory, the space complexity of the algorithm is $O(n^2)$. More efficient algorithms for agglomerative hierarchical clustering methods are represented, for example, in [DE83].

5.4 Experiments

In this section we present how attributes and event sequences can be clustered by similarity. In Subsection 5.4.1 we describe some results on clustering of attributes when similarity between them is computed with the different measures defined in Chapter 3. Then in Subsection 5.4.2 we represent some experimental results on clustering of event sequences as their similarity is determined by the measures defined in Chapter 4.

In our experiments we used the three linkage clustering methods considered in the previous sections. The methods were implemented so that if two clusterings have the same quality value, the one that contains less clusters is chosen as the best one. This choice was made because we wanted the best clusterings chosen by the programs to be as simple as possible, i.e., to contain as few clusters as possible. For selecting the best clustering we used in the programs the clustering measures defined in Section 5.2.

All the clustering experiments were run on a PC with 233 MHz Pentium processor and 64 MB main memory, under the Linux operating system.

5.4.1 Clustering of attributes

One of the reasons for considering attribute similarity was the need for building attribute hierarchies based on the data. Having similarities, or distances between attributes such a hierarchy can be constructed by doing hierarchical clustering on attributes. One possible sketch how an attribute hierarchy could be build efficiently using our distance measures is the following.

Suppose we have computed the distances between all pairs of attributes, and assume A and B are the closest pair. Then we can form a new attribute F as the combination of A and B. This new attribute is interpreted as the union of A and B in the sense that $fr(F) = fr(A) + fr(B) - fr(AB)$.

Suppose we then want to continue the clustering of attributes. The new attribute F represents a new cluster, so we have to be able to compute the distance of F from the other attributes. For this, we need to be able to compute the confidence of the rules $F \Rightarrow D$ for all probes $D \in P$. This confidence is defined as

$$conf(F \Rightarrow D) = \frac{fr((A \vee B)D)}{fr(A \vee B)} = \frac{fr(AD) + fr(BD) - fr(ABD)}{fr(A) + fr(B) - fr(AB)}.$$

Note that if we have computed the frequencies for all subsets of R with sufficiently high frequency, then all the terms in the above formula are known. Thus, we can continue the clustering without having to look at the original data again.

Another possibility for constructing an attribute hierarchy is to use the standard agglomerative hierarchical clustering as we did in our experiments. The hierarchical methods we used were the single, complete and average linkage clustering. The data sets considered in our experiments were the same as in Section 3.5, namely, the Reuters-21578 data and the course enrollment data of the Department of Computer Science at University of Helsinki. For consistency, we considered the same sets of interesting attributes and probe attributes as in our earlier experiments.

Documents and keywords

Figure 5.5 shows four clustering trees for the 14 countries in the Reuters-21578 data set. These clustering trees were produced with the single linkage method by using the internal distance measure $d_{I_{sd}}$ and the external measure $d_{E_{fr},P}$ with the three probe sets P. The clustering tree based on the internal distance $d_{I_{sd}}$ reflects mainly the number of co-occurrence of the keywords in the articles, whereas the clustering trees produced with the external measures weigh the co-occurrence of the keywords with the probe keywords.

All the clustering trees in Figure 5.5 are quite natural, and they correspond mainly to our views of the geopolitical relationships between the countries. The clustering trees are, however, different. The difference in them depend on the distance measure used, and in the case of the external measure the difference also reflect the different probe sets used.

In the clustering tree produced with the internal measure the countries are divided in two main groups: the countries from South America and the others. The best clustering in this tree is, however, the trivial clustering with 14 singleton clusters. The best clustering with the other two clustering methods were the same, even if the clustering trees produced with them were slightly different. This means that according to the clustering measures used, these 14 countries are not similar enough to form larger clusters with a good quality.

The best clustering among the clusterings produced by using the external measure with the economic terms probe set contains two clusters where the smaller of them consists of Canada and USA. In the case of the organizations probe set, the size of the best clustering is also two, but now Ecuador and Venezuela form the smaller cluster. When we use the mixed set of keywords as probes, the best clustering consists of three clusters. The first contains Canada and USA, the second Argentina, China and USSR, and the third the rest of the 14 countries. With other clustering methods the results were nearly the same. Hence, also the results on best clusterings show that the external measure with different probe sets view the set of attributes from different viewpoints.

Students and courses

The clustering trees of the nine courses in the course enrollment data produced with the single linkage method by using the internal measure $d_{I_{sd}}$ and the external measure $d_{E_{fr},P}$ with three probe sets P are shown in Figure 5.6. Using the internal distance measure we get a clustering tree (on the top left of the figure) that reflects how the courses are divided into the three sections at the Department of Computer Science. Distributed Operating Systems, Compilers and Computer Networks are courses from the software section, whereas courses User Interfaces, Database Systems II and Object-Oriented Databases are courses from the section of information systems. The third group of courses, i.e., courses Design and Analysis of Algorithms, String Processing Algorithms, and Neural Networks, belong to the courses from the section of general orientation in computer science. The clustering trees produced with the other two methods were quite similar to this tree.

The best clustering chosen by the single linkage method is the trivial clustering, i.e., the set of 9 singleton clusters. The same result was obtained with the other two methods as well as with all the three methods when we based the clustering on the internal measure $d_{I_{conf}}$. Note that this happened only by chance, and therefore, we cannot expect this kind of a result in general.

The other three clustering trees in Figure 5.6 represent how the nine courses are grouped by using the external measure $d_{E_{fr},P}$ with the set of optional intermediate level courses (top right) the set of mixed advanced level courses (bottom left), and the mixed set of courses from software section (bottom right). The trees are very different, but in every one of them courses from different sections are grouped together. These trees once again confirm our idea on the function of the probe sets in describing the data from different points of view. The clustering trees produced with the other two linkage methods were in the case of the optional intermediate courses and the mixed set of advanced courses rather similar to the trees produced with the single linkage method. However, in the case of the probe set of courses from software section, the clustering trees produced with the three methods were very different from each other. In general, the clustering trees produced by the three methods are at least slightly different from each other. This is only natural, because the definitions of the distance between clusters in the methods are different, and therefore, the clusters chosen to be merged in each phase of the clustering process do not have to be the same with every method.

The best clusterings chosen by the methods when using the external distances contained two or four clusters. With the probe set of optional intermediate courses, the best clustering divides the courses in four clusters. The largest of these clusters contains the courses Neural Networks, Object-Oriented Databases, String Processing Algorithms, and Design and Analysis of Algorithms, and the smallest only the course Computer Networks. The other two clusters have two courses each. With the other two probe sets, the best clustering consists of two clusters, i.e., the seventh clustering is the best one. The division of the courses into the two clusters is, however, different in each of the cases.

As stated earlier, the three clustering methods used can produce from the same set of objects clustering trees with very different structures. Figure 5.7 represents one such situation. In the figure there are three different clustering trees for the nine courses produced by using the external measure with the probe set of compulsory intermediate level courses. Until the fourth clustering, all the methods proceed similarly, but then the way how the clusters are merged begin to differ. Also the best clusterings of courses chosen by the methods are different. In the case of the average linkage clustering the best clustering consists of three clusters, of which the smallest contains only the course Compilers. Using the other two linkage methods, the best clustering is formed by two clusters. The contents of the clusters are, however, very different.

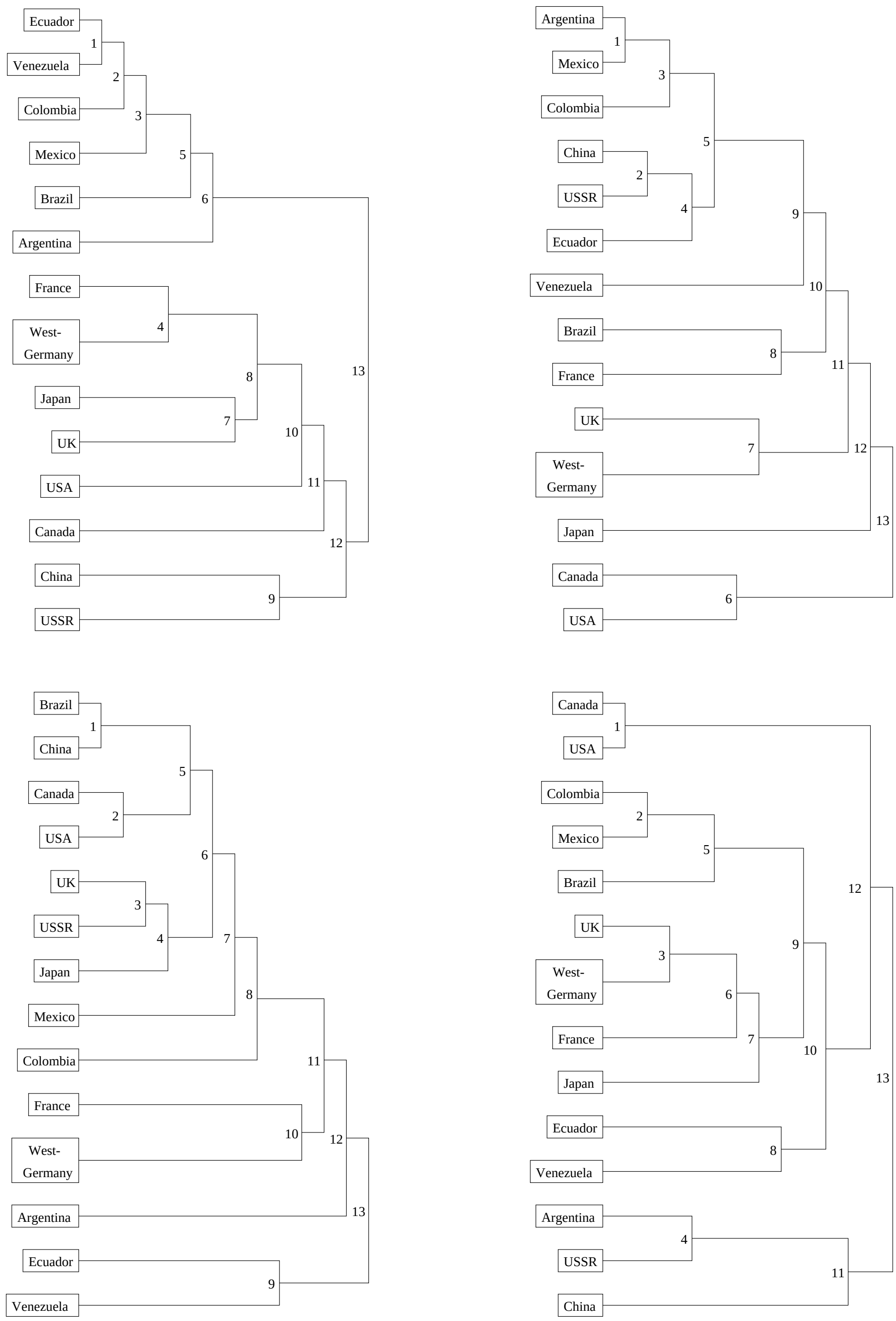

Figure 5.5: Clustering trees of 14 countries produced with single linkage method by using $d_{I_{sd}}$ (top left) as well as $d_{E_{fr},P}$ with the economic terms probe set (top right), and the organizations probe set (bottom left) and the mixed probe set (bottom right).

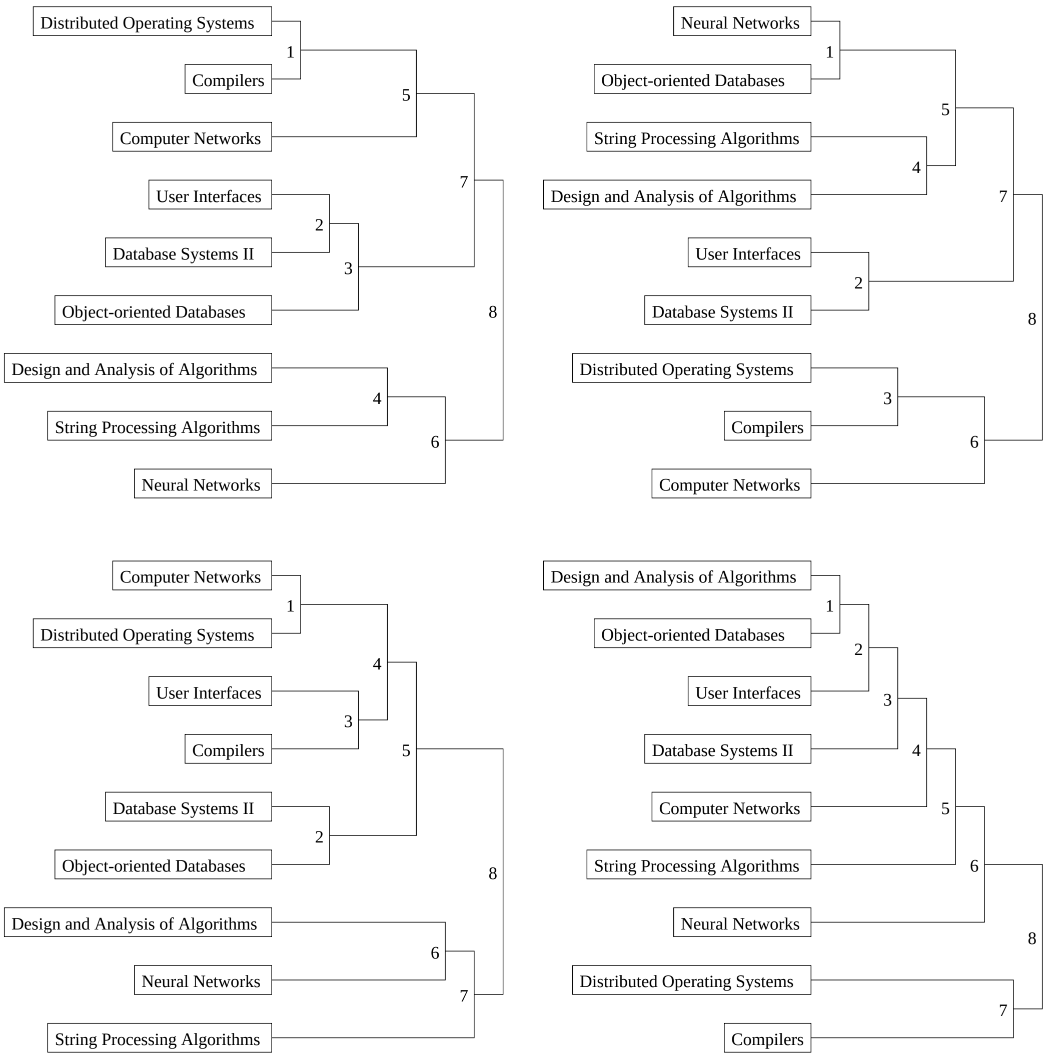

Figure 5.6: Clustering trees of nine advanced level courses produced with single linkage method by using the internal measure $d_{I_{sd}}$ (top left) and the external measure $d_{E_{fr},P}$ with the probe sets of optional intermediate level courses (top right), mixed advanced level courses (bottom left), and courses of software section (bottom right).

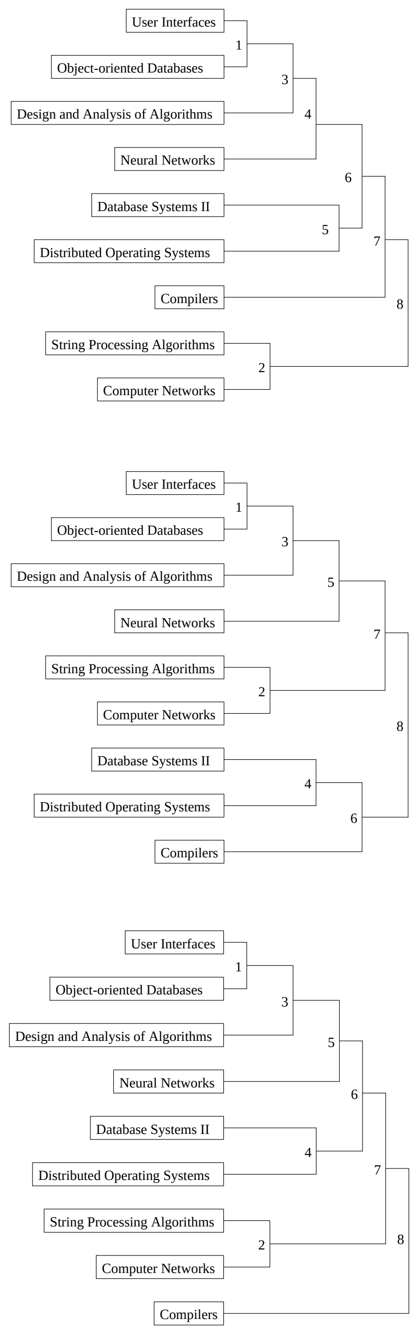

Figure 5.7: Clustering trees of nine advanced level courses produced with single linkage (top), complete linkage (center) and average linkage methods (bottom) when the used distance measure was $d_{E_{fr},P}$ with the probe set of compulsory intermediate level courses.

5.4.2 Clustering of event sequences

There are several reasons why we need clustering of event sequences. For the first, assume that we want to make a query to a collection of event sequences. In order to find efficiently all sequences similar to the query, we need an index of the sequences in the collection. Methods for building such indexes for time series and other numerical sequences has been proposed in [AFS93, BYÖ97, BÖ97, FRM93]. In the case of event sequences, we can construct such an index, for example, by clustering the sequences with respect to their similarity.

On the other hand, in analyzing sequential data one interesting problem is to find a way to predict an occurrence of an event of a given type in a sequence. This problem can also be considered as finding explanations for what usually happens before an event of a given type occurs. For example, in telecommunication network monitoring we could use such information to detect a probable occurrence of a severe fault, early enough to prevent it, or at least to correct it quickly. We could try to solve this prediction problem, for example, by searching from the sequence all *episode rules* [MTV95, MTV97] that tell us what is the probability of an occurrence of an event of the given type, if some other kind of events are known to have occurred. Another possible solution is to search for typical situations that precede occurrences of an event of the given type. Such situations can be found, for example, by clustering the sequences preceding those occurrences of events within a given time period.

In clustering of event sequences we can use the standard agglomerative hierarchical clustering. The hierarchical methods we used in our experiments were the single, complete and average linkage methods. For consistency, the experiments on clustering of event sequences were done with the same sets of sequences, namely, the sets of telecommunication alarm sequences and the sets of sequences of WWW page requests, as in Section 4.4. In the following we present some results on clustering of event sequences, but not on event type sequences. The reason for this choice was that the clusterings of event type sequences were often the same or at least very similar to the clusterings of event sequences.

Telecommunication alarm data

Figure 5.8 shows three clustering trees for the sets of alarm sequences preceding occurrences of events of type 1400, 1660 and 7272. The trees were produced by the single linkage method using the edit distances between event sequences with the alphabet-weighted operation costs. As we mentioned in Section 4.4, it is only a chance that all the sets considered here contain nine event sequences.

The best clusterings of the sequences preceding an alarm of type 1400 or type 7272 consist of three clusters, whereas the trivial clustering of nine clusters was chosen as best of the clusterings of the sequences preceding an alarm of type 1660. The choice of the best clustering in the case of the alarm type 1660 is

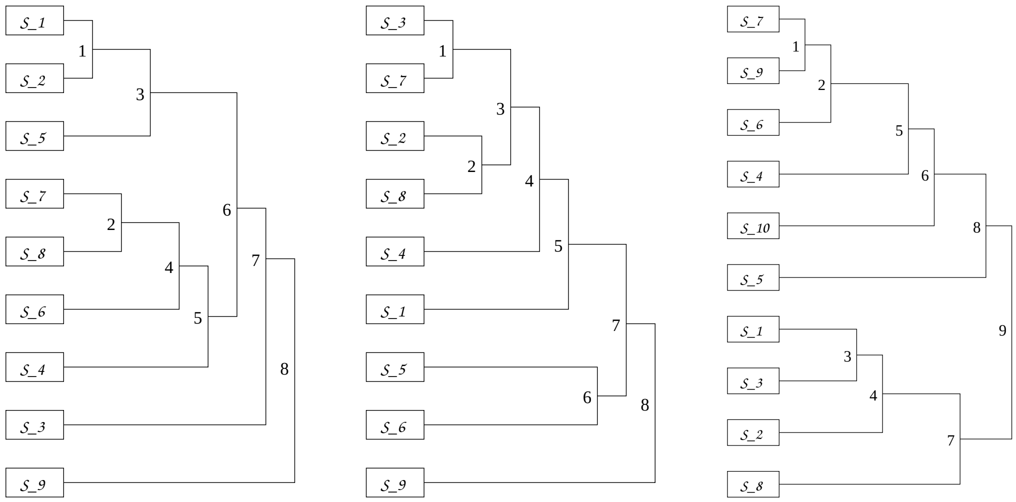

Figure 5.8: Clustering trees of three sets of alarm sequences produced with the single linkage clustering method: of those preceding an alarm of type 1400 (left), those preceding an alarm of type 1660 (center) and those preceding an alarm of type 7272 (right).

still natural, since the sequences in that set are very different from each other. In general, our experiments showed that in the best clusterings all the event sequences within a cluster have at least one event of a common type.

In Section 4.4 we found that the edit distances with unit and alphabet-weighted costs can differ a lot. This leads to an assumption that also the clustering trees resulting from the distances with unit operation costs and with alphabet-weighted costs should be different. This assumption is confirmed by our experiments. For example, in Figure 5.9 we have six different clustering trees of the sequences prededing occurrences of an alarm of type 1400. From the figure we can see that the clustering trees produced with a same clustering method, but with different operation costs are not similar to each other. This tells us that by the choice of operation costs we can influence quite a lot on whether two sequences are clustered together or not. Another interesting thing that we can observe from the figure is that the clustering trees produced with the three clustering methods are different from each other by using unit operation costs, but rather similar to each other, if we used alphabet-weighted operation costs. Note that the clustering tree produced with the single linkage method by using unit costs is a good example of chaining of clusters.

Because the structure of the clustering trees are not similar with unit and alphabet-weighted operation costs, also the best clusterings by using different costs differ a lot from each other. In the case of unit operation costs, the best clusterings contain either five or six clusters, whereas in the case of alphabet-

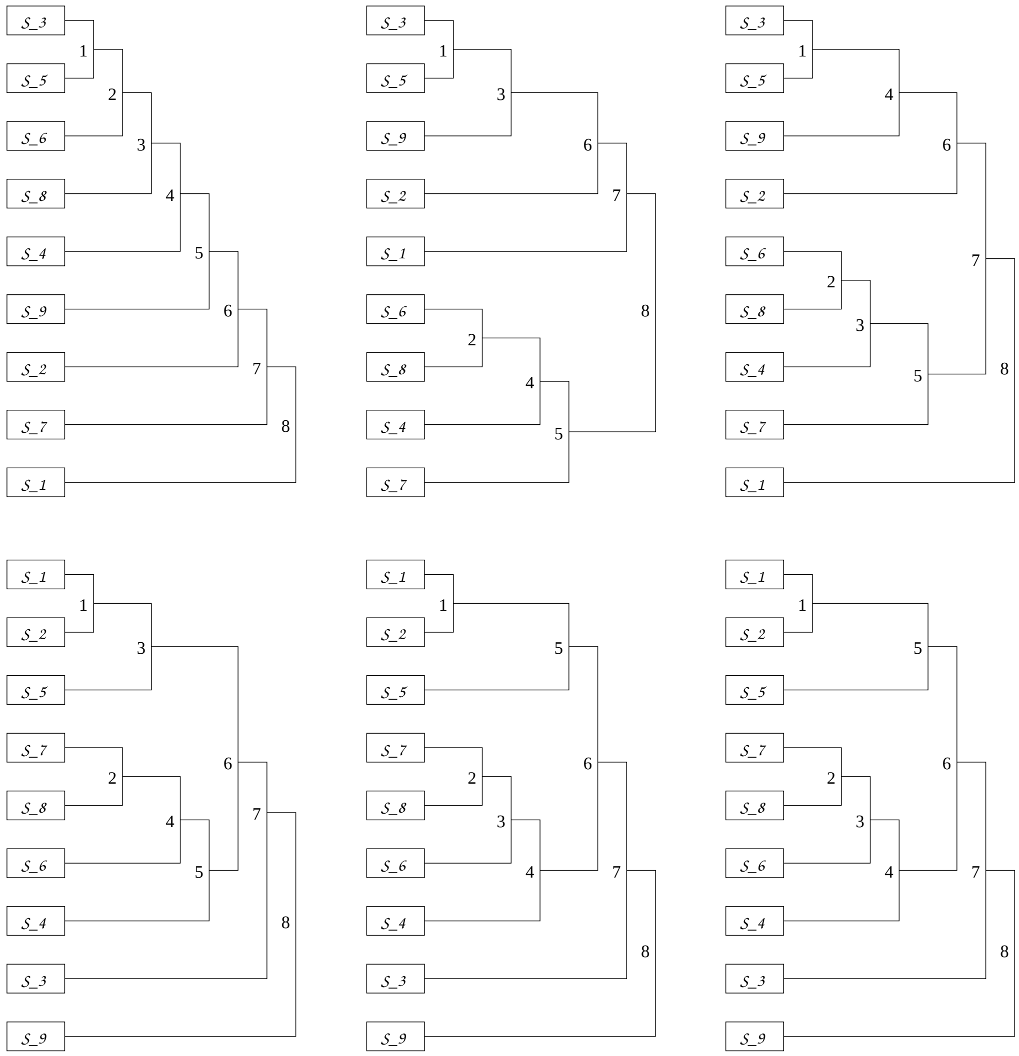

Figure 5.9: Clustering trees of alarm sequences preceding an alarm of type 1440 by using unit (top row) and alphabet-weighted costs (bottom row), produced with the single (left), complete (center) and average (right) linkage methods.

weighted operation costs they contain only three or four clusters. In addition to the number of clusters, the best clusterings differ also in what event sequences are grouped together. However, the best clusterings by using the same operation costs, but different clustering methods are quite similar to each other.

By using unit operation costs, the best clustering with the single linkage method consists of five clusters, of which four are singleton clusters. Also the average linkage methods gives as best a clustering with five clusters. These clusters are, however, quite different from the clusters of the single linkage method. The complete linkage method chooses as best a clustering with six clusters. These clusters are almost the same as with the average linkage method: only the cluster of the event sequences $\mathcal{S}_4, \mathcal{S}_6$ and $\mathcal{S}_8$ is divided in two clusters with the complete linkage method.

In the case of alphabet-weighted costs, we can notice two interesting phenomenona in the clustering trees of Figure 5.9. The structures of the clustering trees produced with the complete and average linkage methods are exactly the same, but the best clusterings chosen by the methods differ from each other. The best clustering with the complete linkage method consists of four clusters and with the average linkage method of three clusters. On the other hand, the clustering trees of the single and average linkage method are a bit different, but the best clustering of them is the same. This example and some of our other experiments show that, if two clustering trees have similar structure, the best clusterings chosen by the different methods can be different, and even if the structures are different, the best clusterings chosen by the methods can be the same.

Sequences of WWW page requests

As in Section 4.4, we considered in the case of the WWW log data the sets of sequences preceding references to the CSUH project page, HeCSE reseach page and KDD link page by using the alphabet-weighted operation costs. In Figure 5.10 there are the clustering trees for these three sets of event sequences

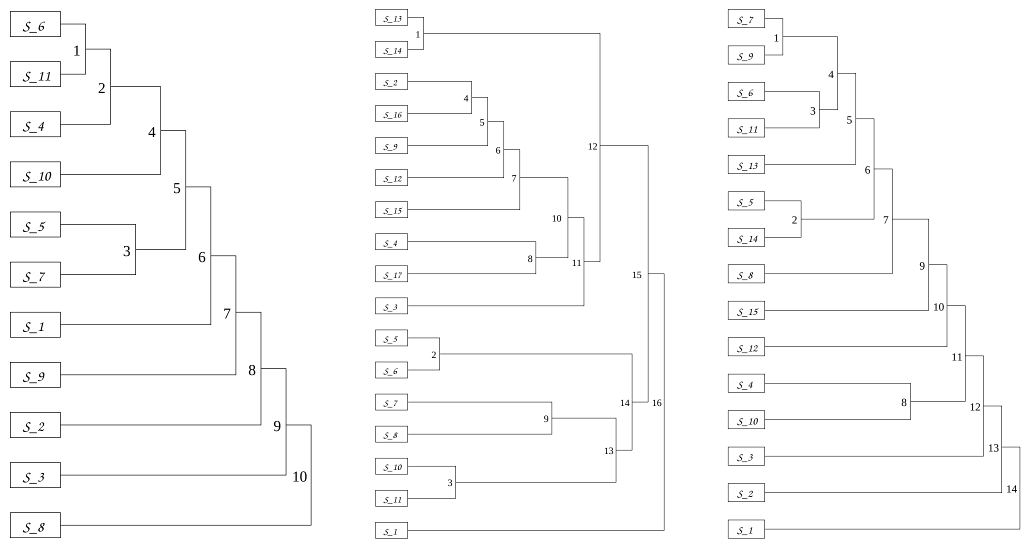

Figure 5.10: Clustering trees of the three sets of event sequences, those preceding a reference to the CSUH project page (left), those preceding a reference to the HeCSE research group page (center) and those preceding a reference to the KDD link page (right) produced with the single linkage clustering method.

produced with the single linkage method. The best clusterings in the cases of the CSUH project page and HeCSE research page have 8 clusters and in the case of the KDD link page 9 clusters. All the best clusterings are natural for the set of event sequences considered.

As in the case of alarm sequences, we wanted see how using different operation costs influence the clusterings. The results for clustering of event sequences preceding references to the CSUH project page with different clustering method and different operation costs are given in Figure 5.11. With both operation

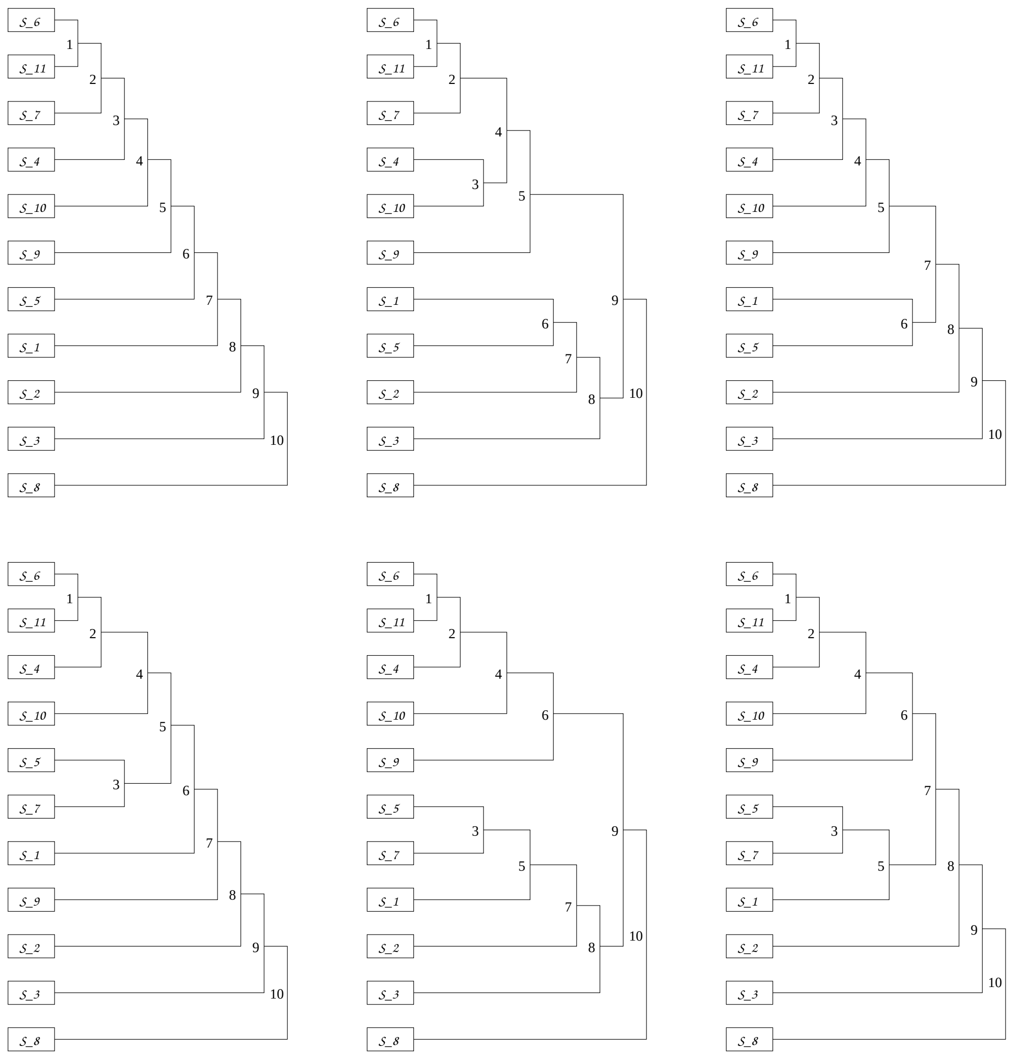

Figure 5.11: Clustering trees of event sequences preceding a reference to the CSUH project page with unit (top row) and alphabet-weighted costs (bottom row), produced with single (left), complete (center) and average (right) linkage methods.

costs, the different clustering methods produced different clustering trees, even though the differences in the tree structures were not always large. The structures of the clustering trees by using a same method but different costs, however, differ quite a lot. This can once again be explained with the fact that by using one type of operation costs two event sequences can be found very similar, and by using another type of operation costs very dissimilar.

The best clusterings with each method contain 9 clusters in the case of unit operation costs and 8 clusters in the case of alphabet-weighted costs. In addition to the number of the clusters, the methods agree with this data set also on which sequences are grouped together. Even though all these results are very natural, they cannot be generalized to any other set of sequences, as was seen, for example, above in the experiments with the alarm sequences (see Figure 5.9).

6 Conclusions

Similarity is an important concept for advanced retrieval and data mining applications. In this thesis we have discussed the problem of defining a similarity or distance notion between objects, and especially in the case of binary valued attributes and event sequences.

We started by considering in Chapter 2 what similarity between objects is and where similarity notions are needed. We defined similarity in terms of a complementary notion of distance and described properties that we expect every distance measure to have. In our opinion, a distance measure should be a metric, or at least a pseudometric. It should also be easy and efficient to compute. Furthermore, such a measure should be natural, and it should capture the true similarity between objects.

In Chapter 3 we presented various ways of defining similarity between binary valued attributes. We started by considering the traditional internal measures of similarity. The value of an internal measure of similarity between two attributes is purely based on the values in the columns of those two attributes. Such measures are in several cases useful but, unfortunately, they cannot reflect certain kinds of similarity. Therefore, we moved on to discuss external measures of attribute similarity.

We introduced an external similarity measure that determines the distance between two attributes by considering the values of a selected set of other attributes. The selected attributes are called probe attributes. We gave experimental results with internal and external measures of attribute similarity on two real-life data sets: the Reuters-21578 newswire data and the course enrollment data of the Department of Computer Science at the University of Helsinki. The results of our experiments showed clearly that the internal and external measures truly describe different aspects of the data. Also using various probe sets gave different similarity notions. This is, however, as it should be: the probe set defines the point of view from which similarity is judged.

After that, in Chapter 4, we studied how similarity between event sequences could be determined. Similarity between numerical sequences has been studied widely, but as far as we know, we have been the first ones to consider similarity between sequences of (event type, occurrence time) -pairs. Our main intuition in defining similarity between event sequences was that it should somehow reflect the amount of work that is needed to convert one event sequence to another. We formalized this notion as edit distance between sequences, and showed that such a measure can be efficiently computed using dynamic programming. We gave experimental results on two real-life data sets: telecommunication alarm data and a log of WWW page requests. The results showed that our definition of distance between event sequences produces an intuitively appropriate notion of similarity. We also studied what kind of an influence associating different costs to the edit operations used in transformations has, and found that with various costs we get different notions of similarity.

Finally, in Chapter 5 we described one possibility of using the similarity notions described in the earlier chapters: clustering of attributes and event sequences by similarity. We presented three standard agglomerative hierarchical clustering methods used in our experiments. We also defined a set of three clustering measures: the distance, the tightness and the quality of clustering. These measures were then used in selecting the best clustering from a hierarchy of clusterings. In our experiments we used four real-life data sets, the two data sets of binary valued attributes and the two sets of event sequences considered in the earlier chapters. The results showed that with the hierarchical methods we can produce natural clusterings of the sets of attributes and event sequences. The difference in the clustering trees reflect the different similarity notions used. The hierarchies of clusterings were, however, different depending on the clustering method and the similarity notion used.

Many interesting problems remain open. Considering attribute similarity, one of these questions is semiautomatic probe selection, i.e., how we could provide guidance to the user in selecting a proper set of probe attributes. We should also examine how the proposed variations of the external measure influence distance values. Furthermore, we should make more experiments to determine the usability of external distances in various application domains. Studying the use of attribute hierarchies in rule discovery and extending the external measure for distance between attribute values are also worth investigating.

The event sequence similarity should be developed further, too. An important problem is to examine more thoroughly influence of different values of the parameters to edit distances. Moreover, we need to extend the edit operation set with a substitution of events and to examine how this change influences distance values. This extension would mean that we make use of attribute similarity in determining similarity between event sequences. Further experimentation is also needed to determine the usability of edit distance in various application domains. Finally, considering other types of notions than edit distance in defining similarity between event sequences might be useful.

References

[AFS93] R. Agrawal, C. Faloutsos, and A. Swami. Efficiency similarity search in sequence databases. In D. B. Lomet, editor, *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms (FODO'93)*, pages 69 – 84, Chicago, IL, USA, October 1993. Springer-Verlag.

[Aho90] A. V. Aho. Algorithms for finding patterns in strings. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, pages 255 – 400. Elsevier Science Publishers B.V (North-Holland), Amsterdam, The Netherlands, 1990.

[AHV95] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison-Wesley Publishing Company, Reading, MA, USA, 1995.

[AIS93] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In P. Buneman and S. Jajodia, editors, *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'93)*, pages 207 – 216, Washington, DC, USA, May 1993. ACM.

[ALSS95] R. Agrawal, K.-I. Lin, H. S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In U. Dayal, P. M. D. Gray, and S. Nishio, editors, *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, pages 490 – 501, Zürich, Switzerland, September 1995. Morgan Kaufmann.

[AMS+96] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. Fast discovery of association rules. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 307 – 328. AAAI Press, Menlo Park, CA, USA, 1996.

[And73] M. R. Anderberg. *Cluster Analysis for Applications.* Academic Press, New York, NY, USA, 1973.

[APWZ95] R. Agrawal, G. Psaila, E. L. Wimmers, and M. Zaït. Querying shapes of histories. In U. Dayal, P. M. D. Gray, and S. Nishio, editors, *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, pages 502 – 513, Zürich, Switzerland, September 1995. Morgan Kaufmann.

[AS95] R. Agrawal and R. Srikant. Mining sequential patterns. In P. S. Yu and A. L. P. Chen, editors, *Proceedings of the Eleventh International Conference on Data Engineering (ICDE'95)*, pages 3 – 14, Taipei, Taiwan, March 1995. IEEE Computer Society Press.

[Bas89] M. Basseville. Distance measures for signal processing and pattern recognition. *Signal Processing*, 18(4):349–369, December 1989.

[BMS97] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In J. M. Peckman, editor, *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'97)*, pages 265 – 276, Tucson, AZ, USA, May 1997. ACM.

[BÖ97] T. Bozkaya and M. Özsoyoğlu. Distance-based indexing for high-dimensional metric spaces. In J. M. Peckman, editor, *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'97)*, pages 357 – 368, Tuscon, AZ, USA, May 1997. ACM.

[BYÖ97] T. Bozkaya, N. Yazdani, and M. Özsoyoğlu. Matching and indexing sequences of different lengths. In F. Golshani and K. Makki, editors, *Proceedings of the Sixth International Conference on Information and Knowledge Management (CIKM'97)*, pages 128 – 135, Las Vegas, NV, USA, November 1997. ACM.

[CPZ97] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. In M. Jarke, M. Carey, K. R. Dittrich, F. Lochovsky, P. Loucopoulos, and M. A. Jeusfeld, editors, *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB'97)*, pages 426 – 435, Athens, Greece, August 1997. Morgan Kaufmann.

[CR94] M. Crochemore and W. Rytter. *Text Algorithms*. Oxford University Press, New York, NY, USA, 1994.

[DE83] W. H. E. Day and H. Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. Report F 121, Technische Universität Graz und Österreichische Computergesellschaft, Institute für Informationsverarbeitung, Austria, July 1983.

[DGM97] G. Das, D. Gunopulous, and H. Mannila. Finding similar time series. In H. J. Komorowski and J. M. Zytkow, editors, *Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'97)*, pages 88 – 100, Trondheim, Norway, June 1997. Springer-Verlag.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley Inc., New York, NY, USA, 1973.

[DJ76] R. Dubes and A. K. Jain. Clustering techniques: The user's dilemma. *Pattern Recognition*, 8:247 – 260, 1976.

[DMR97] G. Das, H. Mannila, and P. Ronkainen. Similarity of attributes by external probes. Report C-1997-66, Department of Computer Science, University of Helsinki, Finland, October 1997.

[DMR98] G. Das, H. Mannila, and P. Ronkainen. Similarity of attributes by external probes. In R. Agrawal, P. Stolorz, and G. Piatetsky-Shapiro, editors, *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD'98)*, pages 23 – 29, New York, NY, USA, August 1998. AAAI Press.

[EN89] R. Elmasri and S. B. Navathe. *Fundamentals of Database Systems.* Addison-Wesley Publishing Company, Reading, MA, USA, 1989.

[FL85] D. Fisher and P. Langley. Approaches to conceptual clustering. In A. K. Joshi, editor, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI-85)*, pages 691 – 697, Los Angeles, CA, USA, August 1985. Morgan Kaufmann.

[FPSSU96] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining.* AAAI Press, Menlo Park, CA, USA, 1996.

[FRM93] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. Report CS-TR-3190, Department of Computer Science, University of Maryland, MD, USA, December 1993.

[FS96] R. Fagin and L. Stockmeyer. Relaxing the triangle inequality in pattern matching. Report RJ 10031, IBM Research Division, Almaden Research Center, San Jose, CA, USA, June 1996.

[GK79] L. A. Goodman and W. H. Kruskal. *Measures of Association for Cross Classifications.* Springer-Verlag, Berlin, Germany, 1979.

[GK95] D. Q. Goldin and P. C. Kanellakis. On similarity queries for time-series data: Constraint specification and implementation. In U. Montanari and F. Rossi, editors, *Proceedings of the 1st International Conference on Principles and Practice of Constraint Programming (CP'95)*, pages 137 – 153, Cassis, France, September 1995. Springer-Verlag.

[Gus97] D. Gusfield. *Algorithms on Strings, Trees and Sequences. Computer Science and Computational Biology.* Cambridge University Press, New York, NY, USA, 1997.

[HCC92] J. Han, Y. Cai, and N. Cercone. Knowledge discovery in databases: an attribute-oriented approach. In L.-Y. Yuan, editor, *Proceedings of the Eighteenth International Conference on Very Large Data Bases (VLDB'92)*, pages 547 – 559, Vancouver, Canada, August 1992. Morgan Kaufmann.

[HKM+96] K. Hätönen, M. Klemettinen, H. Mannila, P. Ronkainen, and H. Toivonen. Knowledge discovery from telecommunication network alarm databases. In S. Y. W. Su, editor, *12th International Conference on Data Engineering (ICDE'96)*, pages 115 – 122, New Orleans, LA, USA, February 1996. IEEE.

[JCH95] I. Jonassen, J. F. Collins, and D. G. Higgins. Finding flexible patterns in unaligned protein sequences. *Protein Science*, 4(8):1587 – 1595, 1995.

[JD88] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data.* Prentice-Hall, Englewood Cliffs, NJ, USA, 1988.

[JMM95] H. V. Jagadish, A. O. Mendelzon, and T. Milo. Similarity-based queries. In A. Y. Levy, editor, *Proceedings of the Fourteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'95)*, pages 36 – 45, San Jose, CA, USA, May 1995. ACM.

[KA96] A. J. Knobbe and P. W. Adriaans. Analysing binary associations. In E. Simoudis, J. W. Han, and U. Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pages 311 – 314, Portland, OR, USA, August 1996. AAAI Press.

[Ket97] A. Ketterlin. Clustering sequences of complex objects. In D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD'97)*, pages 215 – 218, Newport Beach, CA, USA, August 1997. AAAI Press.

[KJF97] F. Korn, H.V. Jagadish, and C. Faloutsos. Efficiently supporting ad hoc queries in large datasets of time. In J. M. Peckman, editor, *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'97)*, pages 289 – 300, Tuscon, AZ, USA, May 1997. ACM.

[KL51] S. Kullbach and R. A. Leibler. On information theory and sufficiency. *Annals of Mathematical Statistics*, 22:79 – 86, 1951.

[KR90] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* John Wiley Inc., New York, NY, USA, 1990.

[Kul59] S. Kullbach. *Information Theory and Statistics.* John Wiley Inc., New York, NY, USA, 1959.

[Lai93] P. Laird. Identifying and using patterns in sequential data. In K. P. Jantke, S. Kobayashi, E. Tomita, and T. Yokomori, editors, *Proceedings of the 4th International Workshop on Algorithmic Learning Theory*, pages 1 – 18, Berlin, Germany, 1993. Springer-Verlag.

[LB97] T. Lane and C. E. Brodley. Sequence matching and learning in anomaly detection for computer security. In T. Fawcett, editor, *AAAI'97 Workshop on Artificial Intelligence Approaches to Fraud Detection and Risk Management*, pages 43 – 49, Providence, RI, USA, July 1997. AAAI Press.

[Lew97] D. Lewis. The Reuters-21578, Distribution 1.0. http://www.research.att.com/~lewis/reuters21578.html, 1997.

[Mie85] O. S. Miettinen. *Theoretical Epidemiology.* John Wiley Inc., New York, NY, USA, 1985.

[MKL95] R. A. Morris, L. Khatib, and G. Ligozat. Generating scenarios from specifications of repeating events. In *Proceedings of the Second International Workshop on Temporal Representation and Reasoning (TIME'95)*, Melbourne Beach, FL, USA, April 1995. IEEE Computer Society Press.

[MR92] H. Mannila and K.-J. Räihä. *Design of Relational Databases.* Addison-Wesley Publishing Company, Wokingham, United Kingdom, 1992.

[MR97] H. Mannila and P. Ronkainen. Similarity of event sequences. In R. Morris and L. Khatib, editors, *Proceedings of the Fourth International Workshop on Temporal Representation and Reasoning (TIME'97)*, pages 136 – 139, Daytona, FL, USA, May 1997. IEEE Computer Society Press.

[MS68] G. Majone and P. R. Sanday. On the numerical classification of nominal data. Research Report RR-118, AD 665006, Graduate School of Ind. Administration, Carnigie-Mellon University, Pittsburgh, PA, USA, 1968.

[MT96] H. Mannila and H. Toivonen. Discovering generalized episodes using minimal occurrences. Research Report C-1996-12, Department of Computer Science, University of Helsinki, Finland, March 1996.

[MTV95] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovering frequent episodes in sequences. In U. M. Fayyad and R. Uthurusamy, editors, *Proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD'95)*, pages 210 – 215, Montreal, Canada, August 1995. AAAI Press.

[MTV97] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259 – 289, 1997.

[Nii87] I. Niiniluoto. *Truthlikeness*. Reidel Publishing Company, Dordrecht, The Netherlands, 1987.

[OC96] T. Oates and P. R. Cohen. Searching for structure in multiple streams of data. In L. Saitta, editor, *Proceedings of the Thirteenth International Conference on Machine Learning (ICML'96)*, pages 346 – 354, Bari, Italy, July 1996. Morgan Kaufmann.

[PR87] L. Pitt and R. E. Reinke. Polynomial-time solvability of clustering and conceptual clustering problems: The agglomerative-hierarchical algorithm. Report UIUCDCS-R-87-1371, Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA, September 1987.

[PR88] L. Pitt and R. E. Reinke. Criteria for polynomial-time (conceptual) clustering. *Machine Learning*, 2:371 – 396, 1988.

[PSF91] G. Piatetsky-Shapiro and W. J. Frawley, editors. *Knowledge Discovery in Databases*. AAAI Press, Menlo Park, CA, USA, 1991.

[RM97] D. Rafiei and A. Mendelzon. Similarity-based queries for time series data. In J. M. Peckman, editor, *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'97)*, pages 13 – 25, Tuscon, AZ, USA, May 1997. ACM.

[SA95] R. Srikant and R. Agrawal. Mining generalized association rules. In U. Dayal, P. M. D. Gray, and S. Nishio, editors, *Proceedings of the 21st International Conference on Very Large Data Bases*

(VLDB'95), pages 407 – 419, Zürich, Switzerland, 1995. Morgan Kaufmann.

[SBM98] C. Silverstein, S. Brin, and R. Motwani. Beyond market baskets: Generalizing association rules to dependence rules. *Data Mining and Knowledge Discovery*, 2(1):39 – 68, 1998.

[SHJM96] A. Shoshani, P. Holland, J. Jacobsen, and D. Mitra. Characterization of temporal sequences in geophysical databases. In P. Svensson and J. C. French, editors, *Proceedings of the Eight International Conference on Scientific and Statistical Database Management (SSDBM'96)*, pages 234 – 239, Stockholm, Sweden, June 1996. IEEE Computer Society Press.

[SK97] T. Seidl and H.-P. Kriegel. Efficient user-adaptable similarity search in large multimedia databases. In M. Jarke, M. Carey, K. R. Dittrich, F. Lochovsky, P. Loucopoulos, and M. A. Jeusfeld, editors, *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB'97)*, pages 506 – 515, Athens, Greece, August 1997. Morgan Kaufmann.

[SM97] J. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology*. PWS Publishing Company, Boston, MA, USA, 1997.

[Ste94] G. A. Stephen. *String Searching Algorithms*. World Scientific Publishing, Singapore, 1994.

[SVA97] R. Srikant, Q. Vu, and R. Agrawal. Mining association rules with item constraints. In D. Heckerman, H. Mannila, D. Pregibon, and R. Uthurusamy, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD'97)*, pages 67 – 73, Newport Beach, CA, USA, August 1997. AAAI Press.

[Toi96] H. Toivonen. Discovery of frequent patterns in large data collections. PhD thesis, Report A-1996-5, Department of Computer Science, University of Helsinki, Finland, December 1996.

[Ull88] J. D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume I. Computer Science Press, Rockville, MD, USA, 1988.

[Vos91] G. Vossen. *Data Models, Database Languages and Database Management Systems*. Addison-Wesley Publishing Company, Reading, MA, USA, 1991.

[WJ96] D. A. White and R. Jain. Algorithms and strategies for similarity retrieval. Technical Report VCL-96-101, Visual Computing Laboratory, University of California, San Diego, CA, USA, July 1996.

[YK58] G. U. Yule and M. G. Kendall. *An Introduction to the Theory of Statistics*. Charles Griffin & Company Ltd., London, UK, 1958.

[YÖ96] N. Yazdani and Z. M. Özsoyoğlu. Sequence matching of images. In P. Svensson and J. C. French, editors, *Proceedings of the Eight International Conference on Scientific and Statistical Database Management (SSDBM'96)*, pages 53 – 62, Stockholm, Sweden, June 1996. IEEE Computer Society Press.