Department of Computer Science
Series of Publications A
Report A-2000-1

Attribute, Event Sequence, and Event Type Similarity Notions for Data Mining

Pirjo Moen

University of Helsinki
Finland

Department of Computer Science
Series of Publications A
Report A-2000-1

Attribute, Event Sequence, and Event Type Similarity Notions for Data Mining

Pirjo Moen

To be presented, with the permission of the Faculty of Science of the University of Helsinki, for public criticism in Auditorium XII, University Main Building, on February 25th, 2000, at 12 o'clock noon.

University of Helsinki
Finland

Contact information

Postal address:
Department of Computer Science
P.O.Box 26 (Teollisuuskatu 23)
FIN-00014 University of Helsinki
Finland

Email address: postmaster@cs.Helsinki.FI

URL: http://www.cs.Helsinki.FI/

Telephone: +358 9 1911

Telefax: +358 9 191 44441

ISSN 1238-8645
ISBN 951-45-9102-X
Computing Reviews (1998) Classification: H.2.8, H.3.3, I.2.6
Helsinki 2000
Helsinki University Printing House

Attribute, Event Sequence, and Event Type Similarity Notions for Data Mining

Pirjo Moen

Department of Computer Science
P.O. Box 26, FIN-00014 University of Helsinki, Finland
Pirjo.Moen@cs.Helsinki.FI, http://www.cs.Helsinki.FI/Pirjo.Moen/

PhD Thesis, Series of Publications A, Report A-2000-1
Helsinki, February 2000, vi + 190 + 9 pages
ISSN 1238-8645, ISBN 951-45-9102-X

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 three kinds of similarity notions: similarity between binary attributes, similarity between event sequences, and similarity between event types occurring in 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 describe all important types of similarity. Therefore, we introduce a new attribute similarity measure that takes into account the values of other attributes in the relation. 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. This model is based on the idea that a similarity notion should reflect how much work is needed in transforming an event sequence into another. We formalize this notion as an edit distance between sequences. Then we show how the resulting measure of distance can be efficiently computed using a form of dynamic programming, and also give some experimental results on two real-life data sets.

As the third case of similarity notions, we study how similarity between types of events occurring in sequences could be defined. Intuitively, two event types are similar if they occur in similar contexts. We show different possibilities for how a context of an event can be extracted from a sequence. Then we discuss ways of defining similarity between two event types by using sets of the contexts of all their occurrences in given sequences. Results of experiments on the event type similarity with different measures are described on both synthetic and real-life data sets.

Computing Reviews (1998) Categories and Subject Descriptors:
H.2.8 Database Applications: Data mining
H.3.3 Information Search and Retrieval
I.2.6 Learning

General Terms:
Algorithms, Experimentation, Theory

Additional Key Words and Phrases:
Data mining, Knowledge discovery, Similarity, Distance, Relational data, Binary attributes, Event sequences, Event types in sequences

Acknowledgements

To begin with, I would like to thank my advisor, Professor Heikki Mannila, for his guidance and encouragement during my doctoral studies. Many of the ideas in this thesis were developed together with him. With some parts of the thesis I also had the pleasure to work with Professor Gautam Das. Furthermore, I am deeply grateful to Dr. A. Inkeri Verkamo, Dr. Mika Klemettinen, Docent Hannu Toivonen, and Professor Dimitrios Gunopulos for their insightful comments that have improved this thesis. M.A. Marina Kurtén helped me with the language, for that I am thankful.

This work has been carried out at the Department of Computer Science of the University of Helsinki. The department, headed by Professors Martti Tienari, Esko Ukkonen, and Timo Alanko, has provided me excellent working conditions during my research. Moreover, I want to express my gratitude to all those persons, that work or have worked with the system support at the department, for their help. The financial support of the Helsinki Graduate School of Computer Science and Engineering (HeCSE), and the Emil Aaltonen Foundation for this work is also gratefully acknowledged.

During my doctoral studies I have been privileged to work and spend time with several people. First I would like to thank all the members of the Data Mining research group at the department. I am also grateful to all my other colleagues, some for their encouragement, and others for their pleasant company and their efforts to give me other things to think about than my thesis. Especially, I want to thank my colleague and friend, M.Sc. Tiina Niklander for spending so many hours discussing with me about both professional and private matters.

I would also want to express my loving and caring thanks to my parents, Raija and Jaakko Ronkainen, for their continuous support and interest in my studies. Neither can I forget my sister Arja and all my friends outside the department for being there for me.

Above all, I want to thank my husband Cristofer for his endless encouragement and patience, not forgetting his ability to ask the "wrong" questions forcing me in new directions. I am also grateful for his support and care for me both in good and bad days during all these years.

Helsinki, January 2000

Pirjo Moen

Contents

Chapter 1

Introduction

The rapid development of computer technology in the last decades has made it possible to easily collect huge amounts of data. For example, a telecommunication network produces 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 exist in the data. A research area in computer science that studies these 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 can 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; see, e.g., [ABKS98, AFS93, BFG99, BÖ97, CPZ97, GIM99, GK95, JMM95, KA96, KE98, KJF97, SK97, WJ96].

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

When discussing similarity and databases, we are often talking about similarity between the objects stored in the database. In market basket data, for example, 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 is, however, another class of similarity notions, i.e., *similarity between (binary) attributes*. In the market basket database setting, we could, for example, define similarities 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 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 all important 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 also takes into account the values of a set of other attributes in the database. We contend that in several cases external measures can give more accurate and useful results than internal measures.

Similarities 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 in data mining to form various kinds of rules, e.g., generalized association rules [HF95, SA95], or characteristic rules and discrimination rules [HCC92]. Often the hierarchy of attributes is supposed to be given by a domain expert. Unfortunately, the domain expertise needed to form such a hierarchy is not always available. Hence, we need a way of computing similarities between attributes and forming such a hierarchy based on these similarities. Another reason for the need of a computable similarity notion between attributes is that it is useful to derive such an attribute hierarchy purely on the basis of the actual data, not by using the *a priori* 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 World Wide Web page request monitoring. Abstractly, such data can be viewed as an *event sequence* that is an ordered collection of *events* from a finite set of *event types*, with each event of the sequence having an occurrence time. In telecommunication network management, for example, event types are the possible error messages, and events are the actual occurrences of errors at certain times. In the same way, in an event sequence of a web access log from a single session of a user the event types are the web pages, and an individual event is a request for a particular page at a particular time.

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. During the last few years, interest to develop methods for knowledge discovery from sequences of events has increased; see, e.g., [AS95, GRS99, GWS98, HKM+96, Lai93, MKL95, MT96, MTV95, MTV97, OC96, WH98, Zha99]. In this thesis we consider two problems concerning similarity and event sequences, namely *similarity between event sequences* and *similarity between event types* occurring in sequences.

First we consider the problem of defining similarity between event sequences. A great deal of work has been done in the area of similarity between time series and other numerical sequences. Note that an event sequence is different from a time series in that a time series describes a variable with a continuous value over time, whereas an event sequence consists of discrete events in time. Thus, the methods for describing similarity between time series are not necessarily suitable for event sequences.

Our approach to event sequence similarity is based on the idea that similarity between event sequences should reflect the amount of work that is needed to transform one event sequence into another. We formalize this notion as an *edit distance* between sequences, and show that the resulting definition of similarity has several appealing features.

A similarity notion between event sequences can be used to build an index of a set of sequences. Such an index could be used, for instance, for efficiently finding all sequences similar to a given pattern sequence. On the other hand, we might 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 type based on the similarities between these sequences.

In this thesis we also study the problem of defining similarity between event types occurring in sequences. In telecommunication network management it would, for example, be interesting to know how similar different alarms are. Our idea of defining similarity between event types is based on the following simple idea: two event types are similar if they occur in similar *contexts*. Abstractly, similarity between two event types is defined by taking sets of all contexts of their occurrences and computing the similarity between these sets. To formalize this intuitive idea we have to answer two questions: (1) What is the context of an occurrence of an event type? and (2) What does it mean that two sets of contexts are similar? We discuss several possibilities for answering these questions and show that even simple answers can yield results that are interesting from the practical point of view.

A similarity notion between event types in sequences is useful in itself, as it provides us with important information about the relationships between the event types in the data set considered. Moreover, similarities between event types can be used in various ways in querying the data set. And of course, similarities between event types are useful in defining similarity between event sequences, especially if we want to look at other things than just equality and inequality of events.

Many of the ideas in this thesis have been developed together with Professor Heikki Mannila, but also together with Professor Gautam Das. Theory of attribute similarity was originally presented in [DMR98] (preliminarily discussed in [DMR97]). The basics for the theory remains in this thesis, but experiments have been reworked. Theory of event sequence similarity was published in [MR97], although it has been somewhat refined here. Ideas of event type similarity described in [MM99][1] remain here, but more have been developed for this thesis. Experiments have also been reworked to the latter two articles.

The rest of this thesis is organized as follows. First, in Chapter 2 we discuss some general properties and uses of similarity notions in data mining. Then, in Chapter 3 we describe some possible similarity measures for binary attributes in relations. In Chapter 4 we represent the main characteristics of event sequences and define similarity between event sequences as the edit distance between them. After that, in Chapter 5, we discuss ways of defining similarity between event types occurring in sequences. Finally, in Chapter 6, we make some concluding remarks and discuss future work.

[1] Note that in 1998 my last name changed from Ronkainen to Moen.

Chapter 2

Similarity notions and their uses

We start by discussing the meaning and uses of similarity notions. Formally such notions are defined using different similarity measures. Therefore, in this chapter we also describe some general properties that we expect a similarity measure to have.

Similarity is an important concept in many research areas; for example, in biology, computer science, linguistics, logic, mathematics, philosophy and statistics, a great deal of work has been done on similarity issues. 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 for such regularities, it is usually not enough to consider only equality or inequality of data objects. Instead, we need to consider how similar, or different 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 (or distance) 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. The following are some typical examples of such applications.

- Market basket data contains a great deal of valuable information about customer behavior in terms of purchased products. Information about products with similar selling patterns can, for example, be useful in planning marketing campaigns and promotions, in product pricing, or in the 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 computational biology the most important primitive operation is a comparison of sequences, where the idea is to find which parts of the sequences are alike and which parts differ. The user can be interested in finding out how similar two DNA sequences of the same length are, for example, or if there are 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 about telecommunication alarm data. The same kind of questions could also be posed about any sequential data, a WWW page request log, for example.

- From financial time series data a user may be interested in finding, for example, stocks that had a large price fluctuation last week, or identifying companies whose stock prices have similar patterns of growth. Same kind of queries one could pose about any set of time series.

- In image databases it might be interesting to retrieve all such images in the database that are similar to the query image with respect to certain colors or shapes, for example.

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

How similarity between objects is defined, however, largely depends on the type of the data. The objects considered in data mining are often complex, and they are described by a different number of different kinds of features. It is, for instance, clear that similarity between binary attributes is determined differently from similarity between images or sounds. Neither can we define similarity between biosequences exactly in the same way as similarity between time series. On the other hand, on a single 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. Neither should similarity between alarm types in telecommunication data be determined with the same similarity notion as similarity between sequences of alarms.

The definition 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, by comparing the results given by these measures, choose the one that best suits our purposes.

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

Definition 2.1 Let $\mathbf{O}$ be a set of objects, and d a measure of a distance between objects in the set $\mathbf{O}$. The measure d is called a *metric*, if it satisfies the following conditions for all objects γ_i, γ_j and γ_k in the set $\mathbf{O}$:

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. $d(\gamma_i, \gamma_k) \leq d(\gamma_i, \gamma_j) + d(\gamma_j, \gamma_k)$. □

Ideally, a distance measure between objects should be a metric. The first condition of Definition 2.1 says that a distance measure d should always have a non-negative value, which is a very natural requirement. The same conclusion holds good for the third condition, which states that a distance measure d should be *symmetric*.

The second condition of Definition 2.1 states that if the distance between two objects is zero, then these objects should be identical, and vice versa. This requirement is quite natural but, unfortunately, it may be too restrictive in some cases. It can happen that according to some measure a distance between two objects is zero, for example, even if the objects are distinct. Then we have the following.

Definition 2.2 Let $\mathbf{O}$ be a set of objects, and d a measure of a distance between objects in the set $\mathbf{O}$. The measure d is called a a *pseudometric*, if it satisfies the conditions

1. $d(\gamma_i, \gamma_j) \geq 0$

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

3. $d(\gamma_i, \gamma_j) = d(\gamma_j, \gamma_i)$
4. $d(\gamma_i, \gamma_k) \leq d(\gamma_i, \gamma_j) + d(\gamma_j, \gamma_k)$

for all objects γ_i, γ_j and γ_k in the set $\mathbf{O}$. □

The conditions 1, 2 and 4 of Definition 2.2 are the same as in the definition of a metric. The only difference between these definitions is that according to a pseudometric the distance of an object to itself is always zero (the condition $2'$), but even a distance between two distinct objects can be zero. In such a case, these two objects are, 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.

The fourth condition in Definitions 2.1 and 2.2 states that a distance measure should satisfy the *triangle inequality*. The need for this property may not be obvious. Consider, however, the problem of searching from a large set of objects for all objects similar to an object γ_i. 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 the object γ_i is also 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 could have a case where $d(\gamma_i, \gamma_j)$ and $d(\gamma_j, \gamma_k)$ are both small, but still the distance $d(\gamma_i, \gamma_k)$ would be large. Such a situation is in many cases 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 larger than 1, but not too large. Because it is not quite certain how useful such a property is in practice, we would like distance measures to satisfy the exact triangle inequality. Still, even measures that do not satisfy the relaxed triangle inequality, can sometimes be used as distance measures. This leads us to the following definition.

Definition 2.3 Let **O** be a set of objects, and d a measure of a distance between objects in the set **O**. The measure d is called a *semimetric*, if it fulfills the conditions

$$\begin{array}{ll} 1. & d(\gamma_i, \gamma_j) \geq 0 \\ 2''. & d(\gamma_i, \gamma_i) = 0, \text{ or } d(\gamma_i, \gamma_j) = 0 \text{ if and only if } \gamma_i = \gamma_j \\ 3. & d(\gamma_i, \gamma_j) = d(\gamma_j, \gamma_i) \end{array}$$

for all objects γ_i, γ_j and γ_k in the set **O**. □

In addition to being a metric, a pseudometric or a semimetric, a distance measure d should be natural in some sense, and it should describe the facets of the data that are thought to be interesting. Moreover, the measure should be easy and efficient to compute. If the size of the object set is reasonable, an algorithm that is quadratic in the number of objects can still be 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 the similarity between objects, a distance measure should also capture properly the notion of similarity. This means that if two objects are similar, 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, only the order of the distance values is important, whereas the actual numerical values of the measures are irrelevant. Two distance measures can then be said to behave similarly if they preserve the same order of the distance values; that is, d and $d^{'}$ agree with each other in the sense that for all γ_i, γ_j and γ_k

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

This means that we can multiply or divide the distance values by any constant without modifying the properties of the measure. If this condition does not hold good, the two measures do not describe the data in the same way, and therefore, they give a different view of the data.

In the following chapters we consider similarity between objects in three particular cases. First, in Chapter 3 we give some measures for defining similarity between binary attributes. How similarity between event sequences could be determined we represent in Chapter 4. And finally, in Chapter 5 we describe ways of defining similarity between event types occurring in sequences.

Chapter 3

Similarity between attributes

One important type of data considered in data mining are relations of data objects with several attributes. Special cases of such relations are those where all the attributes are binary-valued. From such a relation one has usually searched for similarities between the stored data objects. In this chapter, however, we study how to define similarity between the binary attributes in such a relation. We consider two basic approaches to attribute similarity. An *internal measure* of similarity between two attributes is based purely on the values of those two attributes, not on any other attributes in the relation. An *external measure*, on the contrary, also takes into account the values of all or some of the other attributes in the relation. We also study how similarities between attributes obtained with different measures are related to each other, and show how these similarities can be used, for example, to build attribute hierarchies.

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 of similarity. Algorithms for computing attribute similarity measures are given in Section 3.4. Section 3.5 represents the results of our experiments with various data sets and various measures. Finally, in Section 3.6 we discuss the relationships between the various attribute similarity measures. A 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 data 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 use 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 multiset of *m-tuples* called rows. Given a row $\mathtt{t}$ in the relation r, the value of an attribute A_i in the row $\mathtt{t}$ 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 in the relation r by $|r| = n$. Figure 3.1 presents an example of such a relation. □

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

Example 3.1 In a market basket data attributes represent different products such as *beer, potato chips, milk,* and *mustard.* Each row in the relation represents a shopping basket of a customer in a supermarket. If a customer bought just *beer* and *chips*, the row describing his/her shopping basket has values $\mathtt{t}[beer] = 1$ and $\mathtt{t}[chips] = 1$, and a value 0 for all 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, sausage*, and *milk*, and that customer 1 just purchased *chips.* The number of rows in 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, however, attributes like the quantity and the price of the products purchased could also 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 categories: economic subjects, exchanges, organizations, people, and places. For example, keywords associated with an article titled "Bahia Cocoa Review" are *cocoa, El Salvador, USA,* and *Uruguay*, and with an article titled "Six killed in South African gold mine accident" *gold* and *South Africa.* A total of 19 716 articles out of 21 578 have at least one associated keyword. In our examples and experiments, the number of rows in the Reuters data set is, therefore, considered to be 19 716 rows. □

Row ID	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
$\mathtt{t}_1$	1	0	0	0	0	1	0	1	0	0
$\mathtt{t}_2$	1	1	1	1	0	1	0	0	1	1
$\mathtt{t}_3$	1	0	1	0	1	0	0	1	1	0
$\mathtt{t}_4$	0	0	1	0	0	1	0	1	1	1
$\mathtt{t}_5$	0	1	1	1	0	0	1	0	1	1
...										
$\mathtt{t}_{1000}$	1	0	1	1	0	1	0	1	0	1

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

Customer	chips	mustard	sausage	beer	milk	Pepsi	Coke
$\mathtt{t}_1$	1	0	0	0	0	0	0
$\mathtt{t}_2$	0	1	1	0	0	0	0
$\mathtt{t}_3$	1	0	0	0	1	0	0
$\mathtt{t}_4$	1	0	0	1	0	0	1
$\mathtt{t}_5$	0	1	1	0	1	0	0
$\mathtt{t}_6$	1	0	0	1	1	0	1
$\mathtt{t}_7$	0	1	1	0	0	1	0
$\mathtt{t}_8$	1	0	0	0	1	1	0
$\mathtt{t}_9$	0	1	1	1	0	0	1
$\mathtt{t}_{10}$	1	0	0	1	0	0	0
$\mathtt{t}_{11}$	0	1	1	0	1	1	0
$\mathtt{t}_{12}$	0	1	1	0	1	0	0

Figure 3.2: An example of market basket data.

We use letters from the beginning of the alphabet like A and B to denote attributes, and letters from the end of the alphabet like X and Y to denote sets of attributes. An attribute set $X = \{A, \overline{B}, C\}$ 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 the letter r, and rows of relations by the letter $\mathtt{t}$.

In a relation there can be hundreds, or even thousands of attributes, but for each row typically only a few attributes have a value 1, 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 the 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	Purchases
$\mathtt{t}_1$	{chips}
$\mathtt{t}_2$	{mustard, sausage}
$\mathtt{t}_3$	{chips, milk}
$\mathtt{t}_4$	{chips, beer, Coke}
$\mathtt{t}_5$	{mustard, sausage, milk}
$\mathtt{t}_6$	{chips, beer, milk, Coke}
$\mathtt{t}_7$	{mustard, sausage, Pepsi}
$\mathtt{t}_8$	{chips, milk, Pepsi}
$\mathtt{t}_9$	{mustard, sausage, beer, Coke}
$\mathtt{t}_{10}$	{chips, beer}
$\mathtt{t}_{11}$	{mustard, sausage, milk, Pepsi}
$\mathtt{t}_{12}$	{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.

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 the Reuters-21578 collection viewed in a form of sets of keywords associated with the articles.

Typically, we are not interested in every row of the relation at the 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 binary attributes, and r a relation over R. A boolean expression θ, which is constructed from atomic formulae of the form "$\mathtt{t}[A] = 1$" and "$\mathtt{t}[A] = 0$", is called a *selection condition* on the rows of a relation r. A *subrelation* of r that consists of the rows satisfying 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 $\mathtt{t}[A] = 1$, is denoted by r_A, and the number of rows in it by $|r_A|$. Similarly, we denote by $r_{\overline{A}}$ the subrelation of r where the attribute A has value 0, and the number of rows in this subrelation by $|r_{\overline{A}}|$. □

Example 3.3 A subrelation of *beer* buyers r_{beer} in the example market basket data in Figure 3.2 consists of four rows, i.e., the rows $\mathtt{t}_4, \mathtt{t}_6, \mathtt{t}_9$, and $\mathtt{t}_{10}$ of the relation r. On the other hand, a subrelation $r_{\overline{milk}}$ of non-*milk* buyers consists of rows $\mathtt{t}_1, \mathtt{t}_2, \mathtt{t}_4, \mathtt{t}_7, \mathtt{t}_9$ and $\mathtt{t}_{10}$, and the number of rows in it, therefore, is six. □

Example 3.4 A subrelation r_{USA} consists of 12 541 rows where keyword *USA* occurs in the Reuters-21578 data set, and a subrelation $r_{El\ Salvador}$ of 11 rows with keyword *El Salvador*. On the other hand, a subrelation $r_{\overline{Switzerland}}$ consists of 19 502 rows (note that if the rows without any keywords were also considered, the number of rows in the subrelation $r_{\overline{Switzerland}}$ would be 21 364 rows). □

The number of rows in a subrelation indicates the number of rows satisfying the given selection condition. Often we are not interested in the absolute number of rows but rather would like to consider the relative number of the rows, i.e., 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 (relative) *frequency* of the subrelation r_θ is denoted by

$$fr(\theta, r) = \frac{|r_\theta|}{|r|}.$$

If the relation r is clear from the context, we may write $fr(\theta)$. Additionally, we use the abbreviation $fr(A)$ for $fr(\mathtt{t}[A] = 1)$, and $fr(ABC)$ for $fr(\mathtt{t}[A] = 1 \wedge \mathtt{t}[B] = 1 \wedge \mathtt{t}[C] = 1)$. Similarly, $fr(\mathtt{t}[A] = 0)$ is denoted as $fr(\overline{A})$ and $fr(\mathtt{t}[A] = 0 \wedge \mathtt{t}[B] = 1 \wedge \mathtt{t}[C] = 0)$ as $fr(\overline{A}B\overline{C})$. We are usually interested only in the presence of attributes, i.e., the cases where $\mathtt{t}[A] = 1$, 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 $fr(beer) = 4/12 = 0.33$, and the frequency of non-*milk* buyers $fr(\overline{milk}) = 6/12 = 0.50$. □

Example 3.6 In the Reuters-21578 data set the most frequent keyword is *USA* with the frequency of 0.6360. For the other keywords in Example 3.4 we have frequencies $fr(El\ Salvador) = 0.0006$, and $fr(\overline{Switzerland}) = 0.9891$. □

In Definition 3.3 we presented that the frequency can be defined for both the presence and the absence of the attribute, i.e., for the cases of $\mathtt{t}[A] = 1$ and $\mathtt{t}[A] = 0$, 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 absent attributes can be computed from the frequencies of present attributes, i.e., $fr(\overline{A}) = 1 - fr(A)$, of course assuming that we know all the needed frequencies $fr(A)$. 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 over 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 simply write $fr(X \cup Y)$ and $conf(X \Rightarrow Y)$. □

The frequency of an association rule is a measure of 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 also matches 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 above to be a set of attributes. However, in this thesis we only need to consider 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 the frequency of *sausage* buyers is $fr(sausage) = 0.50$, and the frequency of *mustard* buyers $fr(mustard) = 0.50$. The frequency of the rule "*sausage* $\Rightarrow$ *mustard*" is also 0.50 and the confidence of the rule is $0.50/0.50 = 1.00$. This means that every customer that bought *mustard*, also bought *sausage*, and vice versa.

The frequency of *chips* buyers in the same relation is $fr(chips) = 0.50$ and the frequency of *beer* buyers $fr(beer) = 0.33$. 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 who bought *beer* also bought *chips*. On the other hand, the frequency of customers buying *chips* but not *beer* is $fr(\mathtt{t}[beer] = 0 \wedge \mathtt{t}[chips] = 1) = 0.25$, and the frequency of customers buying *beer* but not *chips* is $fr(\mathtt{t}[beer] = 1 \wedge \mathtt{t}[chips] = 0) = 0.08$. So, the confidence of the rule "$chips \Rightarrow \overline{beer}$" is $0.25/0.50 = 0.50$ and the confidence of the rule "$beer \Rightarrow \overline{chips}$" is $0.08/0.33 = 0.24$. □

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 $0.0185/0.6360 = 0.0291$. The frequency of the rule "$El\ Salvador \Rightarrow grain$" is 0.0001 and the confidence of the rule is $0.0001/0.0006 = 0.1667$. 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 the same article, but as much as 16.67 % of the articles talking about *El Salvador* also mention the keyword *grain*.

The frequency of the rule "$\overline{Switzerland} \Rightarrow grain$" is

$$\begin{aligned} fr(grain, \overline{Switzerland}) &= fr(grain) - fr(grain, Switzerland) \\ &= 0.0319 - 0.0001 = 0.0318 \end{aligned}$$

and the confidence of the rule is $0.0318/0.9891 = 0.0322$. On the other hand, the confidence of the rule "$grain \Rightarrow \overline{Switzerland}$" is $0.0318/0.0319 = 0.9969$. 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 a possibility of about 3 % that the keyword *grain* occurs among the keywords of the article. In fact, there is only one article with which both these keywords are associated. □

We will now define similarity between binary attributes. As stated in Chapter 2, we can define similarity between attributes in terms of a complementary notion of a distance between attributes. Then we have the following general definition.

Definition 3.5 Given a set of binary attributes R and a class of all possible relations $\mathcal{R}$ over R, a *distance* measure d between binary attributes is defined as $d : R \times R \times \mathcal{R} \to \mathbb{R}$. Given a relation r over 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 simply $d(A, B)$. $\square$

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

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

Definition 3.6 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 $\mathcal{AI}$. The size of the set $\mathcal{AI}$, i.e., the number of attributes in the set, is denoted by $|\mathcal{AI}|$. $\square$

The selection of interesting attributes depends on the situation and application we are considering. A natural requirement is that the interesting attributes should be intuitively similar. We might think that *juice* and *washing powder*, for example, are not intuitively very similar products in the market basket data. Despite that, in some cases they might still be considered as 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. *Butter, cheese, milk,* and *yogurt*, for example, 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. $\square$

3.2 Internal measures of similarity

An internal measure of similarity between attributes is a measure whose value for two binary attributes A and $B \in R$ is only based on the values of the columns of these attributes. So, such a measure describes 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 fulfill the condition "$\mathtt{t}[A] = 1 \wedge \mathtt{t}[B] = 1$", the value n_{10} the number of the rows that fulfill 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 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 one *degree of freedom.*

There are, of course, numerous ways of defining measures for the strength of association between attributes; see [GK79] for an excellent review 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

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

where n_{ij} represents the observed and $(n_{i.} \cdot n_{.j}/n)$ the expected number of rows with $\mathtt{t}[A] = i \wedge \mathtt{t}[B] = j$ when $n_{i.}$ is the observed number of the rows with $\mathtt{t}[A] = i$ and $n_{.j}$ the observed number of the rows with $\mathtt{t}[B] = j$. 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].

	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.

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 zero. On the other hand, the attributes are considered to be dependent on each other, if the value of the χ^2 test statistic is higher than a *cutoff value* at a given significance level. The cutoff value at the given significance level and with one degree of freedom can, in turn, be obtained from the common table of the significant points of χ^2 available in nearly every book of statistics.

Example 3.11 Consider products *beer* and *milk* in the example market basket data. The contingency table and the table of expected values for these attributes are given in Figure 3.6. With the values in these tables, the χ^2 test statistic is

$$\chi^2 = \frac{(1-2)^2}{2} + \frac{(3-2)^2}{2} + \frac{(5-4)^2}{4} + \frac{(3-4)^2}{4} = 1.50.$$

At the 5 % significance level and with one degree of freedom, the cutoff value of χ^2 is 3.84. Because $1.50 < 3.84$, we cannot reject the independence assumption at this significance level, i.e., the products *beer* and *milk* cannot be said to be significantly dependent on each other at this significance level.

Consider then products *beer* and *Coke* in the same relation. For these products the contingency table and the table of expected values are presented in Figure 3.7. Using these tables, the value of the χ^2 test statistic is

$$\chi^2 = \frac{(3-1)^2}{1} + \frac{(1-3)^2}{3} + \frac{(0-2)^2}{2} + \frac{(8-6)^2}{6} = 8.00.$$

Because $8.00 > 3.84$, i.e., the value of the χ^2 test statistic is higher than the cutoff value, the products *beer* and *Coke* are said to be dependent on each other at the 5 % significance level. □

Example 3.12 Consider then keywords *USA* and *Switzerland* in the Reuters-21578 data. The contingency table and the table of the expected

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

Figure 3.6: The contingency table (a) and the table of expected values (b) for the products *beer* and *milk*.

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

Figure 3.7: The contingency table (a) and the table of expected values (b) for the products *beer* and *Coke*.

values for these keywords are given in Figure 3.8. 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 significance level.

However, for keywords *USA* and *El Salvador* the contingency table presented in Figure 3.9 is the same as the table of the expected values for the numbers of occurrences of these keywords. In this case the value of the χ^2 test statistic is zero, and the keywords can be said to be independent at any significance level. $\square$

The good thing with measuring the 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 should have expected values greater than 5. In the example market basket data, for instance, this causes difficulties. In addition, the total number of rows n considered should be reasonably large.

a)

	Switzerland	$\overline{\text{Switzerland}}$	$\sum$
USA	35	12 506	12 541
$\overline{\text{USA}}$	179	6 996	7 175
$\sum$	214	19 502	19 716

b)

	Switzerland	$\overline{\text{Switzerland}}$	$\sum$
USA	136.12	12 404.88	12 541
$\overline{\text{USA}}$	77.88	7 097.12	7 175
$\sum$	214	19 502	19 716

Figure 3.8: The contingency table (a) and the table of expected values (b) for the keywords *USA* and *Switzerland*.

	El Salvador	$\overline{\text{El Salvador}}$	$\sum$
USA	7	12 534	12 541
$\overline{\text{USA}}$	4	7 171	7 175
$\sum$	11	19 705	19 716

Figure 3.9: The contingency table of the keywords *USA* and *El Salvador*.

Because of the problems with the χ^2 measure, we consider 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 two binary attributes A and $B \in R$, an *internal distance* $d_{I_{sd}}$ between them in a relation r over R is defined as

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

□

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 $\mathrm{t}[A] = 1$ and $\mathrm{t}[B] = 1$. The distance measure $d_{I_{sd}}$ is a complement of the well-known non-invariant 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 $d_{I_{sd}}$ is 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 the frequency of AB is also 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. On the other hand, the attributes are completely dissimilar, i.e., $d_{I_{sd}} = 1$, when $fr(AB) = 0$.

Example 3.13 If we consider the products *beer* and *milk* in the example market basket data, we notice that the frequency of *beer* buyers is $fr(beer) = 0.33$, the frequency of *milk* buyers $fr(milk) = 0.50$, and the frequency of customers buying both *beer* and *milk* is $fr(beer, milk) = 0.08$. Using these values the distance $d_{I_{sd}}$ between *beer* and *milk* is $\frac{0.33+0.50-2\cdot 0.08}{0.33+0.50-0.08} = 0.89$. Hence, according to this measure *beer* buyers do not tend to buy *milk*, and vice versa. In this sense the *beer* buyers behave differently than the *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 $d_{I_{sd}}$ between 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 *El Salvador* is 0.0006, the frequency of the keyword *Switzerland* 0.0109, and the frequency of the keyword *USA* 0.6360. The frequency of the keyword set $\{Switzerland, USA\}$ is 0.0018. With these values we get the distance $d_{I_{sd}}(Switzerland, USA) = 0.9972$. The frequencies $fr(El\ Salvador, USA) = 0.0004$ and $fr(El\ Salvador, Switzerland) = 0$ in their turn give us distances $d_{I_{sd}}(El\ Salvador, USA) = 0.9994$ and $d_{I_{sd}}(El\ Salvador, Switzerland) = 1$. 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. □

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

Definition 3.8 Let A and B be binary attributes in the set R, and $A \Rightarrow B$ and $B \Rightarrow A$ association rules computed from a relation r over R. Then an *internal distance* $d_{I_{conf}}$ between attributes A and B is defined as

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

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 exactly similar have distance 0 and attributes that are completely 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., $fr(A) = fr(B) = fr(AB)$. In the latter case both the confidences are zero, which means that $fr(AB) = 0$, i.e., the attributes A and B never have the value 1 in the same row.

Example 3.15 Consider the buyers of *beer*, *milk* and *Coke* in our example market basket data. The internal distance $d_{I_{conf}}$ between *beer* and *milk* buyers is $d_{I_{conf}}(beer, milk) = (1 - 0.25) + (1 - 0.17) = 1.58$. 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 $d_{I_{conf}}(beer, Coke) = (1 - 0.75) + (1 - 1) = 0.25$. Therefore, the buyers of these two beverages can be said to behave rather similarly. These results are similar to the ones obtained with the distance measure $d_{I_{sd}}$ in Example 3.13. □

Example 3.16 The distance $d_{I_{conf}}$ between keywords *Switzerland* and *USA* is 1.8320, between keywords *El Salvador* and *USA* 1.3327, 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. Recall that the same kind of results were also obtained with the distance measure $d_{I_{sd}}$ in Example 3.14. □

Internal measures represent the traditional way of defining attribute similarity, and they are useful in many applications. Unfortunately, because they are based solely on the values of the two attributes considered, they do not necessarily find all important 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 3.10.

We computed the values of the three internal distance measures considered in this section for the three pairs of products above. These distance values are shown in Table 3.1. According to the χ^2 test statistic the products *milk* and *sausage* are, indeed, statistically independent. The products of the two other pairs are dependent on each other. Knowing this and by looking at the contingency tables in Figure 3.10 we can see that 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 completely 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 similar to the results 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 3.10. 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.

None of the internal measures can view the three types of situations described in Figure 3.10 as reflecting that the two attributes A and B are similar. Still, the similarity between the attributes A and B in each case can be high, if the similarity between them is 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, the values of other attributes than A and B also have an influence on the similarity between these two attributes.

3.3 External measures of similarity

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

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

Distance 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

Table 3.1: The values of three internal distance measures for three pairs of product in the example market basket data.

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 to the behavior of the buyers of 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 can be defined similar if they occur in articles in a similar way with respect to a set of other keywords. Thus, keywords *El Salvador* and *USA* would be deemed similar, if they are associated in 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 Let R be the set of attributes and r a relation over R. A *probe set* $P = \{D_1, D_2, \ldots, D_k\}$ is a subset of the attributes in R. We call these attributes D_i in P *probe attributes.* Given two binary attributes A and $B \in R$, the relation r and the probe set P, an *external measure of similarity* between attributes says that the attributes A and B are similar if subrelations r_A and r_B are similar with respect to P. □

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

We originally wanted to define similarity between two attributes of size $n \times 1$, and now we have reduced 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]:

$$d(g_A \,\|\, g_B) = \sum_x g_A(x) \cdot \log \frac{g_B(x)}{g_A(x)}$$

or a symmetrized version of it: $d(g_A \,\|\, g_B) + d(g_B \,\|\, g_A)$. 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.

3.3.1 Basic measure

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

Definition 3.10 Let R be a set of binary attributes, A and B two attributes in R, P a set of probe attributes, r a relation over R, and $E_{\mathsf{f}}(A, B, D_i)$ a function for measuring how similar the attributes A and B are with respect to a probe attribute $D_i \in P$. An *external distance* $d_{E_{\mathsf{f}},P}$ between the attributes A and B in the relation r is then defined as

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

i.e., as the sum of the values of E_{f} over all the probe attributes $D_i \in P$. $\square$

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

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

where n is the size of the relation r. When summed over all the probes $D_i \in P$, we get a distance measure $d_{E_{\chi^2},P}(A, B) = \sum_{D_i \in P} E_{\chi^2}(A, B, D_i)$. 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 $E_{\mathsf{f}}(A, B, D_i)$

as the difference in the relative frequencies of the probe attribute D_i in the subrelations r_A and r_B. We then have the following.

Definition 3.11 Let A and B be two binary attributes in the set R, r_A and r_B the corresponding subrelations of a relation r over R, P a set of probe attributes, and D_i a probe attribute in the set P. The difference in the frequencies of the probe attribute D_i in the relations r_A and r_B is

$$E_{fr}(A, B, D_i) = \mid \mathit{fr}(D_i, r_A) - \mathit{fr}(D_i, r_B) \mid.$$

Now the external distance between attributes A and B is

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

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

$$d_{E_{fr},P}(A, B) = \sum_{D_i \in P} \mid \mathit{conf}(A \Rightarrow D_i) - \mathit{conf}(B \Rightarrow D_i) \mid.$$

□

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 its value can be zero even if the attributes compared are not identical, i.e., $A \neq B$. This happens when $\mathit{fr}(D_i, r_A) = \mathit{fr}(D_i, r_B)$ with every probe attribute $D_i \in P$. 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 products *milk* and *sausage* in our example market basket data. Assume then that we have a probe set $P = \{beer, Coke, Pepsi\}$. With this probe set, the products *milk* and *sausage* have the external distance

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

The same result can also be 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.

Now consider the products *chips* and *sausage* in the same relation. If we have a probe set $P = \{milk\}$, these products have an external distance

$d_{E_{fr},P}(chips, sausage) = |\,\frac{3}{6} - \frac{3}{6}\,| = 0$. Therefore, the products *chips* and *sausage* are similar in relation to 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 complement products. □

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 internal measures in Example 3.17. Therefore, even such pairs of attributes that according to internal measures are completely different can be found to be highly similar by using the external distance measure $d_{E_{fr},P}$. This result corresponds to the intuition that similarity between attributes A and B can also be due to some other factors that just the information given by the values in the columns A and B.

3.3.2 Variations

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

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

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

   where $p \geq 1$.

2. For each probe attribute D_i we could give a weight $w(D_i)$ describing its significance in the relation r. The external distance between attributes A and B in that way would be

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

3. The probe set P could be generalized to a set of boolean formulas θ_i on attributes. Then the external measure distance between attributes A and B would be

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

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 attributes is a metric or even a pseudometric. Evaluating the exact importance and effect of these variations is, however, left for further study.

3.3.3 Constructing external measures from internal measures

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

One set of such functions is 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 belonging to the set R, an internal measure of distance between attributes can be used to define the 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 $D_i \in P$ are presented as a vector

$$\mathsf{v}_{A,P} = [\, d_{I_\mathsf{f}}(A, D_1), \ldots, d_{I_\mathsf{f}}(A, D_k) \,].$$

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

Example 3.21 Consider products *milk* and *sausage* in the example market basket data. Assume that we have a probe set $P = \{beer, Coke, Pepsi\}$ and that we use the internal distance $d_{I_{sd}}$ for describing relations between the interesting products and the probe attributes. Then, for the product *milk* we have the vector

$$\begin{aligned} \mathsf{v}_{milk,P} &= [\, d_{I_{sd}}(milk, beer), d_{I_{sd}}(milk, Coke), d_{I_{sd}}(milk, Pepsi) \,] \\ &= [\, 0.889, 0.875, 0.714 \,]. \end{aligned}$$

Similarly, for the product *sausage* we have the vector

$$\begin{aligned} \mathsf{v}_{sausage,P} &= [\, d_{I_{sd}}(sausage, beer), d_{I_{sd}}(sausage, Coke), \\ &\quad\; d_{I_{sd}}(sausage, Pepsi)\,] \\ &= [\, 0.889, 0.875, 0.714\,]. \end{aligned}$$

If we now use a measure corresponding to the Manhattan distance as the distance d between two vectors, the external distance between the products *milk* and *sausage* is zero. Therefore, the customers buying *milk* and *sausage* are said to behave similarly with respect to buying *beer*, *Coke* and *Pepsi*. Note that the same result would also be obtained with all the general Minkowski distance measures. □

3.3.4 Selection of probe attributes

In developing the external measure of similarity our goal was that probe attributes 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 of probes produce different similarity measures. This is demonstrated by the experiments described in Section 3.5. Therefore, a proper selection of the probe set is crucial for the usefulness of 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 though the problem of selecting probes is highly dependent on the application domain and the situation considered, we 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 into the probe set all the other attributes except the attributes A and B, i.e., use the set $P = R \setminus \{A, B\}$ as a probe set. This set is probably inappropriate in most cases, especially if the number of attributes in the relation r is high. Another simple way is to select a fixed amount of attributes as probes, the five attributes that have the highest or the lowest frequencies, for example, if either the common or the rare attributes, respectively, would be considered to be good probe attributes. We could also define a threshold for the frequency of attributes and choose the attributes whose frequency is higher than the given threshold as probes.

When the number of attributes in the probe set P grows, the distance between attributes A and B, of course, increases or at least stays the same. If we add one new probe D_{k+1} to the probe set P, we get a new probe set

$Q = P \cup \{D_{k+1}\}$. Using Definition 3.11, the distance between attributes A and B with the probe set Q is

$$d_{E_{fr},Q}(A,B) = d_{E_{fr},P}(A,B) + |\, conf\,(A \Rightarrow D_{k+1}) - conf\,(B \Rightarrow D_{k+1})\,|.$$

Irrespective of the number of probes in the set P, the external distance between the attributes A and B 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 $D_i \in P$ is such an attribute, confidences of the association rules $A \Rightarrow D_i$ and $B \Rightarrow D_i$ are both nearly one. This means that the probe D_i has little effect on the whole distance. An extreme case is when $fr\,(D_i, r) = 1$, because then the confidences above are both exactly 1 and the external distance $d_{E_{fr},\{D_i\}}(A,B) = 0$. Thus, such a probe D_i has no effect at all on the external distance. If the frequency $fr\,(D_i, r)$ is, however, low compared to the frequencies of the attributes A and B, the confidences $conf\,(A \Rightarrow D_i)$ and $conf\,(B \Rightarrow D_i)$ are also low. This again means that the change in the external distance is small. Thus, adding or excluding a probe attribute with a very high or very low frequency typically does not produce dramatic changes in the external distance between attributes.

The probe selection problem can also be considered more formally. Assume, for example, that for some reason we know (or wish) attributes A and B to be more similar than attributes A and C. Then we can try to search for a probe set that implies this fact, and use this probe set (if one exists) to find distances between 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 singletons $P = \{D_i\}$ satisfying $d_{E_{fr},P}(A,B) < d_{E_{fr},P}(A,C)$, or alternatively, the largest set P of probes 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_i satisfying all these constraints, or all the possible probe sets P satisfying them. Algorithms for finding such probe attributes are given in [DMR97].

3.4 Algorithms for computing attribute similarity

In this section we present algorithms for computing both internal and external distances between binary attributes. We also briefly study the time and space complexities of these algorithms.

Algorithms for computing internal distances

For computing the internal distances $d_{I_{sd}}$ and $d_{I_{conf}}$ between attributes we can use Algorithms 3.1 and 3.2. The input of these trivial algorithms is a set $\mathcal{AI}$ of interesting attributes. In addition to this, Algorithm 3.1 requires a frequency $fr(A)$ of each $A \in \mathcal{AI}$, and a frequency $fr(AB)$ for all pairs of attributes A and B in the set $\mathcal{AI}$ computed from a certain relation r over R. Algorithm 3.2, on the other hand, requires confidences of the association rules $A \Rightarrow B$ in a relation r for all $A, B \in \mathcal{AI}$ as the additional input, instead of the frequencies of attribute pairs. The output of both algorithms are all the pairwise internal distances between the attributes in the set $\mathcal{AI}$ in the relation r.

Algorithm 3.1 Internal distance $d_{I_{sd}}$ between attributes

Input: A set $\mathcal{AI}$ of interesting attributes, frequencies of all the attributes in the set $\mathcal{AI}$ and frequencies of all the pairwise sets of the attributes in the set $\mathcal{AI}$, computed from a relation r.
Output: Pairwise internal distances between the attributes in the set $\mathcal{AI}$ in the relation r.
Method:

```
for all attribute pairs (A, B) where A and B ∈ AI do
    calculate d_{I_sd}(A, B);
od;
output the pairwise internal distances d_{I_sd}(A, B);
```

Algorithm 3.2 Internal distance $d_{I_{conf}}$ between attributes

Input: A set $\mathcal{AI}$ of interesting attributes, and confidences of the association rules $A \Rightarrow B$ for all attributes $A, B \in \mathcal{AI}$, computed from a relation r.
Output: Pairwise internal distances between the attributes in the set $\mathcal{AI}$ in the relation r.
Method:

```
for all attribute pairs (A, B) where A and B ∈ AI do
    calculate d_{I_conf}(A, B);
od;
output the pairwise internal distances d_{I_conf}(A, B);
```

Algorithm for computing external distances

Algorithm 3.3 presents a method for computing the external distance $d_{E_{fr},P}$ between attributes. The input of the algorithm are a set $\mathcal{AI}$ of interesting attributes and a set P of probe attributes. Frequencies $fr(A)$ of the interesting attributes, and frequencies $fr(AD_i)$, for each

$A \in \mathcal{AI}$ and $D_i \in P$ computed from a relation r, are also given as input to the algorithm. Another possibility would be to give as input, instead of the frequencies above, frequencies of all probe attributes $D_i \in P$ in all subrelations r_A, when $A \in \mathcal{AI}$. This is the same as giving the confidences of the rules $A \Rightarrow D_i$, for each $A \in \mathcal{AI}$ and $D_i \in P$ computed from a relation r. The algorithm first computes the value of the function $E_{fr}(A, B, D_i)$ for each probe $D_i \in P$, and then adds it to the distance value already computed. The output of the algorithm is the set of pairwise external distances between the attributes in the set $\mathcal{AI}$ in the relation r.

Algorithm 3.3 External similarity $d_{E_{fr},P}$ between attributes

Input: A set of interesting attributes $\mathcal{AI}$, a set P of probe attributes, frequencies of the interesting attributes in the set $\mathcal{AI}$ and frequencies of all pairs (interesting attribute, probe attribute) in a relation r.

Output: Pairwise external distances between the attributes in the set $\mathcal{AI}$ in the relation r.

Method:

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

Complexity considerations

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

For computing the values of the internal measure $d_{I_{sd}}$ we need frequencies of all attributes $A \in \mathcal{AI}$ and also the frequencies $fr(AB)$, where both A and B are in the set $\mathcal{AI}$. There are $|\mathcal{AI}| + \binom{|\mathcal{AI}|}{2}$ such frequencies. On the other hand, for computing the values of the measure $d_{I_{conf}}$ we need a total of $2 \cdot \binom{|\mathcal{AI}|}{2}$ confidence values of association rules $A \Rightarrow B$, where both A and B are in the set $\mathcal{AI}$. When we have $|\mathcal{AI}|$ interesting attributes, Algorithms 3.1 and 3.2 both require $O(|\mathcal{AI}|^2)$ space. Both algorithms compute internal distances between $\binom{|\mathcal{AI}|}{2}$ pairs of attributes. If we assume that computing an internal distance, either $d_{I_{sd}}$ or $d_{I_{conf}}$, between two attributes takes a constant time, the time complexity of both algorithms is also $O(|\mathcal{AI}|^2)$.

For computing the values of the external distance measure $d_{E_{fr},P}$ we need frequencies of all attributes $A \in \mathcal{AI}$ and all pairwise frequencies $fr(AD_i)$, where $A \in \mathcal{AI}$ and $D_i \in P$. There are a total of $|\mathcal{AI}| + |\mathcal{AI}| \cdot |P|$ such frequencies. If we use the confidence values of the association rules $A \Rightarrow D_i$, for each A and D_i, as input, the number of values needed is $|\mathcal{AI}| \cdot |P|$. When there are $|\mathcal{AI}|$ interesting attributes and $|P|$ probe attributes, the space complexity of Algorithm 3.3 is $O(|\mathcal{AI}|\,|P|)$. The algorithm computes distances between $\binom{|\mathcal{AI}|}{2}$ pairs of attributes, and computing an external distance $d_{E_{fr},P}$ between two attributes takes a $|P|$ time. Therefore, the time complexity of Algorithm 3.3 is $O(|\mathcal{AI}|^2\,|P|)$.

3.5 Experiments

In this section we present experiments that we made on similarity between binary attributes. First, in Section 3.5.1 we describe the data sets used in these experiments. The results of our experiments are then represented in Section 3.5.2. All the experiments were run on a PC with a 233 MHz Pentium processor and a 64 MB main memory under the Linux operating system.

3.5.1 Data sets

In our experiments on similarity between attributes we used two real-life data sets: the Reuters-21578 collection of newswire articles [Lew97], and the course enrollment data collected at the Department of Computer Science at the University of Helsinki. Both these data sets resided in flat text files.

Documents and keywords

The Reuters-21578 collection consists of 21 578 news articles from the year 1987. Most of these articles have a few keywords describing their contents. For example, an article titled "National average prices for farmer-owned reserve" has keywords *grain, wheat, corn, barley, oat, sorghum* and *USA*. There are 674 possible keywords, of which only 445

Keyword set name	Interesting keywords
15 countries	Argentina, Australia, Brazil, Canada, China, Cuba, France, Japan, Mexico, New Zealand, South Africa, United Kingdom, USA, USSR, West Germany
17 commodities	cocoa, coffee, copper, corn, cotton, gold, grain, iron-steel, livestock, oilseed, rice, rubber, silver, soybean, sugar, vegetable oil, wheat

Table 3.2: The sets $\mathcal{AI}$ of interesting keywords in the Reuters-21578 data.

occur in the data set considered. The possible keywords are divided in five categories: economic subjects, exchanges, organizations, people and places. A total of 1 862 articles have no keywords at all. We omitted these articles from the data set, which means that in our experiments the data set consists of 19 716 rows, instead of 21 578 rows. One of the articles in this data set has 29 keywords, and the average number of keywords per article is slightly over two.

We selected several sets $\mathcal{AI}$ of interesting attributes, i.e., interesting keywords from this data set of 19 716 rows. In the following we show detailed results obtained for only two of them. The first of the chosen sets $\mathcal{AI}$ of interesting keywords is a set of *15 countries*[1], and the second a set of *17 commodities*. The keywords belonging to these two sets are given in Table 3.2.

For the set of 15 countries we first computed the internal distances $d_{I_{sd}}$ and $d_{I_{conf}}$, and then the external distances $d_{E_{fr},P}$ by using five probe sets. The probe sets of *economic terms*, *energy terms*, *food commodities*, *international organizations* and *mixed terms* are given in Table 3.3. For the set of 17 commodities, though, we computed the internal distances $d_{I_{sd}}$ and $d_{I_{conf}}$, as well as the external distances $d_{E_{fr},P}$ by using three probe sets. One of these probe sets is the set of 15 interesting countries given in Table 3.2, and the two other probe sets are the sets of economic terms and international organizations given in Table 3.3.

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

Probe set name	Probe attributes
economic terms	balance of payments, consumer price index, gross national/domestic product, industrial production index, money supply, reserves, retail sales, trade, unemployment, wholesale price index
energy terms	crude oil, fuel oil, gasoline, heat oil/gas oil, jet & kerosine, naphtha, natural gas, petro-chemicals, propane
food commodities	barley, corn, grain, oat, potato, rice, rye sorghum, soybean, wheat
organizations	EC, GATT, IMF, OECD, OPEC, Worldbank
mixed terms	corn, crude oil, earnings, grain, interest rates, mergers/acquisitions, money/foreign exchange, natural gas, rice, shipping, soybean, trade, wheat

Table 3.3: The sets P of probe attributes in the Reuters-21578 data.

Students and courses

The course enrollment data contains information about 6 966 students of the Department of Computer Science at the University of Helsinki. This data was collected during the years 1989 – 1996. The original data set was transformed into a relation where each row of the relation describes the course enrollments of one student. One first year student has enrolled in the courses *Introduction to Computing, Computer Systems Organization,* and *Programming (Pascal)*, for example. The courses taught at the department are divided into three classes: basic, intermediate and advanced level courses. Further, the advanced level courses are divided by the section to which they belong. This division into courses from sections of general computer science, computer software, and information systems is not, however, very strict. The number of different courses in the whole data set is 173. About one third of the students (2 528 students) has enrolled only in one course, and one student in a total of 33 different courses. The average number of different courses each student has enrolled in is close to five.

Course set name	Interesting courses
nine advanced level courses	Database Systems II, Object-Oriented Databases, User Interfaces, Compilers, Computer Networks, Distributed Operating Systems, Design and Analysis of Algorithms, Neural Networks, String Processing Algorithms
ten different level courses	Computer Systems Organization, Programming (Pascal), Information Systems, Database Systems I, Data Structures, Computers and Operating Systems, Design and Analysis of Algorithms, Distributed Operating Systems, Compilers, Database Systems II

Table 3.4: The sets $\mathcal{AI}$ of interesting courses in the course enrollment data.

Similarly to the case of the Reuters-21578 data set, we selected from the course enrollment data set diverse sets $\mathcal{AI}$ of interesting attributes, i.e., interesting courses. In the following we give results obtained for only two of these sets. The first of the sets consists of *nine advanced level courses*, and the second of *ten different level courses*. The courses belonging to these two sets are given in Table 3.4. Of the nine advanced level courses, the first three are courses from the section of information systems, the next three courses from the section of computer software, and the last three courses of general orientation in computer science. The first three courses of the set of ten different level courses, in turn, are basic level courses, the next three intermediate level courses, and the last four advanced level courses. All the basic and intermediate courses in this set are compulsory courses for every student, whereas each advanced level course in this set is a compulsory course in some section of the department.

For the set of nine advanced level courses we computed the internal distances $d_{I_{sd}}$ and $d_{I_{conf}}$, as well as the external distances $d_{E_{fr},P}$ by using four probe sets. The probe sets of *compulsory intermediate level courses*, *optional intermediate level courses*, *advanced level courses*, and *courses from the section of computer software* are given in Table 3.5. For the set of ten different level courses we computed, in addition to the internal distances $d_{I_{sd}}$ and $d_{I_{conf}}$, the external distances $d_{E_{fr},P}$ by using four probe sets. The first one of these probe sets is a set of nine *mixed courses*, and the other three are its subsets. All these four probe sets are given in Table 3.6.

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

Table 3.5: The sets P of probe attributes used in the experiments with the set $\mathcal{AI}$ of nine advanced level courses.

Probe set name	Probe attributes
mixed courses	Computer-Aided Instruction, Computer Graphics, Computer Networks, Data Communications, Fundamentals of ADP, Introduction to Unix, Programming in C, String Processing Algorithms, User Interfaces
basic level courses among mixed courses	Fundamentals of ADP, Introduction to Unix, Programming in C
intermediate level courses among mixed courses	Computer-Aided Instruction, Computer Graphics, Data Communications
advanced level courses among mixed courses	Computer Networks, String Processing Algorithms, User Interfaces

Table 3.6: The sets P of probe attributes used in the experiments with the set $\mathcal{AI}$ of ten courses from different levels.

3.5.2 Results

We start by presenting some comparisons of the distance values given by the two internal distance measures $d_{I_{sd}}$ and $d_{I_{conf}}$ for our test sets. After that we explain how the internal distances $d_{I_{sd}}$ between attributes are related to the values given by the external distance measure $d_{E_{fr},P}$ with different probe sets P. We also consider how the external distances with different probe sets differ from each other. Finally, we show how the computed distance values can be used for building attribute hierarchies.

All the distances in this section are computed using programs that correspond to the algorithms given in Section 3.4. We give some examples of the actual distance values, but as stated in Chapter 2, these values are often irrelevant. Therefore, what we are more interested in, when comparing different measures, is the order of distance values given by them.

Internal distances

Figure 3.11a shows the distribution of points $(d_{I_{sd}}(A, B), d_{I_{conf}}(A, B))$ of internal distances for all the keyword pairs (A, B) of the chosen 15 countries in the Reuters-21578 data set. Many values of the measure $d_{I_{conf}}$ are near two, which is the maximal value of this measure. This indicates that for the most of the country pairs (A, B), the confidences of the association rules $A \Rightarrow B$ and $B \Rightarrow A$ are both rather low. Similarly, a large fraction of the values of the measure $d_{I_{sd}}$ are close to one. These results are natural, because when we look at the original data, we can see that the chosen countries seldom occur in the same articles. For example, the keyword *Cuba* occurs in the same articles only with three of the other chosen countries, namely with the keywords *Brazil*, *United Kingdom* and *USSR*, and even with them very seldom. The pair of countries that has the lowest value of the measure $d_{I_{conf}}$ is the pair *(USA, USSR)*, whereas the pair *(France, West Germany)* has the shortest distance according to the measure $d_{I_{sd}}$.

We also made a similar comparison of the internal distances between the 17 commodities. The distribution of points $(d_{I_{sd}}(A, B), d_{I_{conf}}(A, B))$ in this set is given in Figure 3.11b. In this case, most of the keyword pairs also have a long distance according to the two internal measures. There are, however, three pairs of commodities that are indicated to be rather similar by both these measures. These pairs of commodities are *(oilseed, soybean)*, *(grain, wheat)*, and *(corn, grain)*. For all these three pairs, the frequencies of the keywords themselves as well as the frequencies of their pairwise combinations are about the same magnitude. This means that

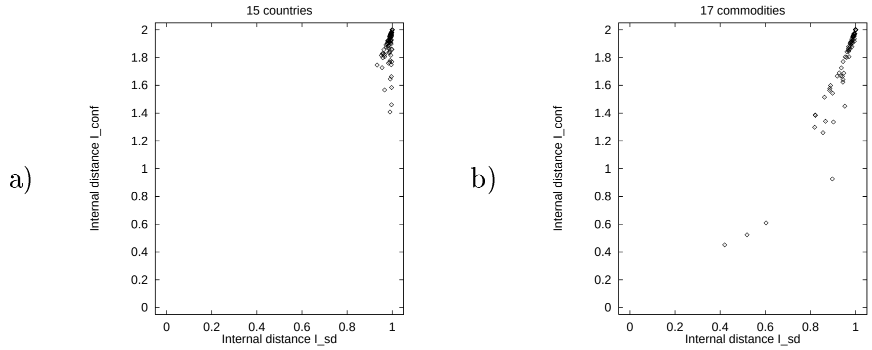


Figure 3.11: Comparison of internal distances $d_{I_{sd}}$ and $d_{I_{conf}}$ between a) 15 countries, and b) 17 commodities.

these pairs of keywords occur rather often in the same articles. According to the measure $d_{I_{conf}}$, the keywords *grain* and *rice* can also be said to be quite similar. What is interesting, is that according to the measure $d_{I_{sd}}$ these two keywords are not particularly similar. An explanation for this result is rather simple. The keyword *grain* occurs much more often in the articles than the keyword *rice*. The frequency $fr(grain, rice)$, however, is about the same as the frequency of the keyword *rice* alone. Therefore, the value of the term $1 - conf(rice \Rightarrow grain)$ in the measure $d_{I_{conf}}$ is nearly zero, and even though $1 - conf(grain \Rightarrow rice)$ is quite high, the distance is still less than one. The measure $d_{I_{sd}}$, on the other hand, describes the symmetric distance of the rows where the keywords *grain* and *rice* occur. Therefore, it takes into account the differences in the frequencies better, and regards the keywords as dissimilar.

Similar experiments were also made with the two sets of interesting courses from the course enrollment data. Figure 3.12a presents the distribution of points $(d_{I_{sd}}(A,B), d_{I_{conf}}(A,B))$ for the nine advanced level courses. In this case, there are no pairs of courses that have a particularly short distance with any of the internal distance measures, neither are the pairs of courses extremely dissimilar. In fact, only the courses *Computer Networks* and *String Processing Algorithms* are determined completely dissimilar by both internal measures $d_{I_{sd}}$ and $d_{I_{conf}}$. With both measures $d_{I_{sd}}$ and $d_{I_{conf}}$, the most similar pair of courses is the pair *(Compilers, Distributed Operating Systems)*. This is a natural result, because these courses belong to the courses from the section of computer software and are, in fact, compulsory courses for all the students in that section. Because the students typically enroll in several courses of the same section, it is also natural that many of the other courses that belong to the same section

a) b)

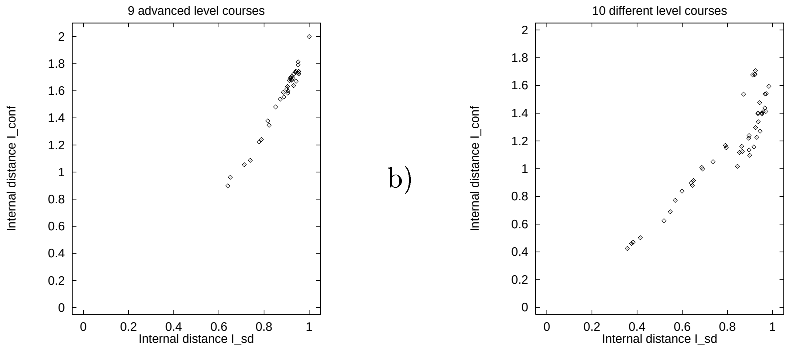


Figure 3.12: Comparison of internal distances $d_{I_{sd}}$ and $d_{I_{conf}}$ between a) nine advanced level courses, and b) ten different level courses.

have a short internal distance. On the other hand, courses that belong to different sections typically have quite a long internal distance, according to both internal measures $d_{I_{sd}}$ and $d_{I_{conf}}$.

For the set of ten courses from different levels, the relationship between the internal distances $d_{I_{sd}}$ and $d_{I_{conf}}$ is shown in Figure 3.12b. Here the pair of courses with the lowest value by both internal distance measures is the pair *(Data Structures, Database Systems I)*. Also some other pairs of courses have quite short distances. Typically, with both measures, courses from the basic and intermediate levels have short internal distances. In this set of ten courses, no courses are completely dissimilar according to either measure. The most dissimilar courses according to the measure $d_{I_{conf}}$ are the courses *Compilers* and *Database Systems II*, which are advanced level courses from different sections. In other cases, the measure $d_{I_{conf}}$ also assigns a high distance value for courses of different sections. Using the measure $d_{I_{sd}}$, however, the pair *(Design and Analysis of Algorithms, Programming (Pascal))* has the longest distance. Similarly, many other pairs where one of the courses is a basic level course and the other an advanced level course have a long internal distance. This shows that the two internal measures stress somewhat different aspects of the data, even though generally they behave rather similarly.

Internal versus external distances

Figure 3.13 describes how the internal distances $d_{I_{sd}}$ between the chosen 15 countries correspond to their external distances given by the measure $d_{E_{fr},P}$ with the five probe sets P in the Reuters-21378 data. The distributions of the points $(d_{I_{sd}}(A,B), d_{E_{fr},P}(A,B))$ in these plots vary a great deal depending on the probe set P used in computing the external distances. The distributions are especially wide with the probe sets of food commodities (Figure 3.13c) and mixed terms (Figure 3.13e).

According to the internal distance measure $d_{I_{sd}}$, the two countries with the shortest distance are *France* and *West Germany*. The pair of countries that has the shortest distance by different external measures $d_{E_{fr},P}$ varies depending on the probe set P used. However, none of the external measures regards *France* and *West Germany* as the most similar pair of countries. According to the internal measure $d_{I_{sd}}$, there are a total of twenty-four pairs of countries that have an internal distance $d_{I_{sd}}(A, B) = 1$. That is, they are completely dissimilar because they never occur in the same articles. However, our experiments show that an external distance between such a pair of countries can be either very long or very short. Keywords *Cuba* and *New Zealand*, for example, are classified by the measure $d_{I_{sd}}$ as completely dissimilar keywords. However, using the probe set of energy terms, their external distance is zero, i.e., they are said to be completely similar. All these results indicate that internal and external measures truly describe different things.

Similar plots of internal and external distances between the 17 commodities are given in Figure 3.14. In this case the plots are also different from each other. With the internal measure $d_{I_{sd}}$, the most similar pair of commodities is the pair *(oilseed, soybean)*. However, with external measures, the most similar pair of commodities is different depending on the probe set P used, and none of them defines *oilseed* and *soybean* as the pair of commodities with the shortest distance. Also the most dissimilar pair of commodities varies depending on the probe set P used. In general, with the economic term probe set, the external distances have very small values, and with the organization probe set the external distances are only slightly higher. These results indicate that, for the pairs (A, B) of commodities, the differences between the confidences $conf(A \Rightarrow D_i)$ and $conf(B \Rightarrow D_i)$, where $D_i \in P$, are small in those two probe sets. Actually, most of the 17 commodities occur only seldom with the chosen economic terms or organizations, and thus, the confidence values are also small. This means that the 17 commodities are similar in the absence of the probe keywords in the sets of economic terms and international organizations. If we, however, are interested in finding more differences between the pairs of commodities in the presence of the probe keywords, we should use the probe set of 15 countries for computing the external distances between the 17 commodities. Namely, with this probe set of 15 countries, the external distances between the 17 commodities vary more than with the other two probe sets.

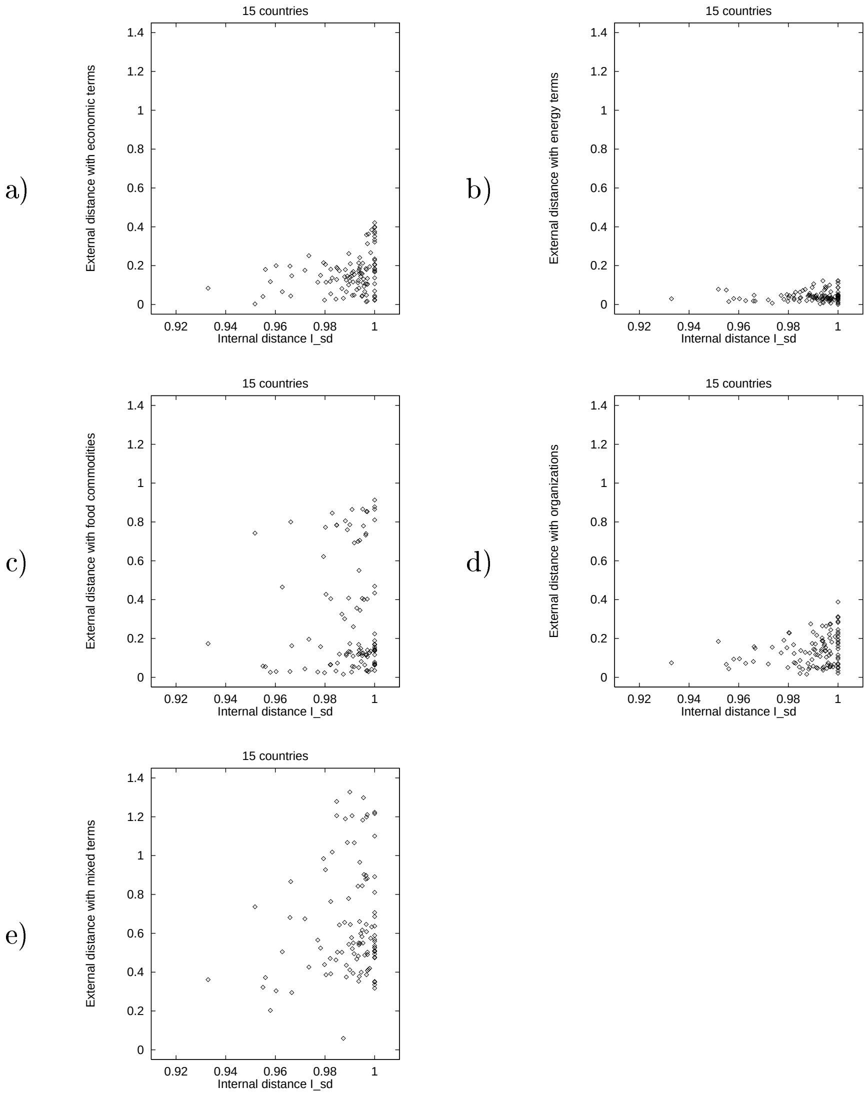


Figure 3.13: Comparison of internal distances $d_{I_{sd}}$ and external distances $d_{E_{fr},P}$ between 15 countries with the probe sets of a) economic terms, b) energy terms, c) food commodities, d) organizations, and e) mixed terms.

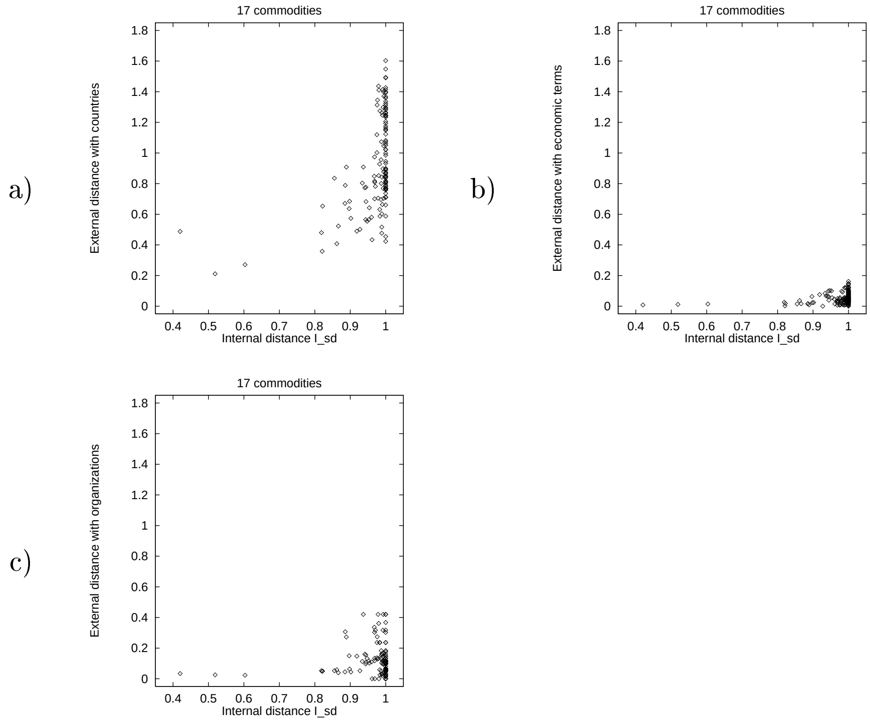


Figure 3.14: Comparison of internal distances $d_{I_{sd}}$ and external distances $d_{E_{fr},P}$ between 17 commodities with the probe sets of a) countries, b) economic terms, and c) organizations.

This can be explained by the fact that some of the commodities occur more often in the same articles with all or some of the 15 countries than other commodities.

Experiments for comparing internal distances $d_{I_{sd}}$ and external distances $d_{E_{fr},P}$ with different probe sets P were also made with the two sets of interesting courses in the course enrollment data set. For the nine advanced level courses the distributions of points $(d_{I_{sd}}(A,B), d_{E_{fr},P}(A,B))$ with the four probe sets P are presented in Figure 3.15. The distributions of the points again show that internal and external measures describe different aspects of the data. The distributions of points in the plots are especially wide when the probe set P used in computing the external distances is either the set of compulsory intermediate courses, or the set of courses from the section of computer software. According to the internal distance measure $d_{I_{sd}}$, the most similar pair of courses is the pair *(Compilers, Distributed Operating Systems)*. However, this pair is not the most similar pair

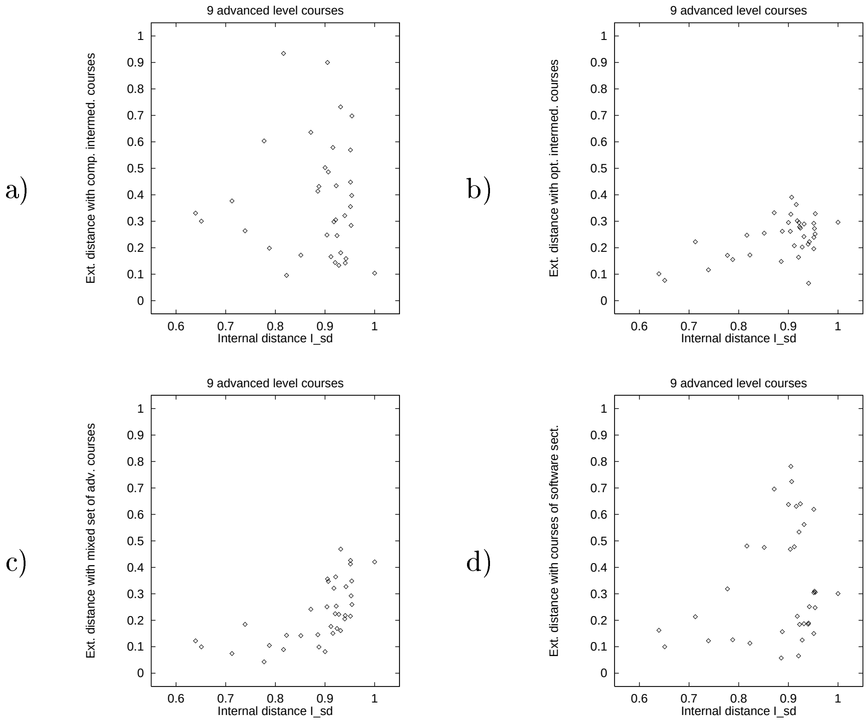


Figure 3.15: Comparison of internal distances $d_{I_{sd}}$ and external distances $d_{E_{fr},P}$ between nine advanced level courses with the probe sets of a) compulsory intermediate level courses, b) optional intermediate level courses, c) advanced level courses, and d) courses from the section of computer software.

of courses with any of the external measures. Similarly, the most dissimilar pair of courses determined by the measure $d_{I_{sd}}$, namely the pair (*Computer Networks*, *String Processing Algorithms*), is not the most dissimilar pair of courses according to any of the external measures. Actually, with the probe set of compulsory intermediate level courses, *Computer Networks* and *String Processing Algorithms* are among the most similar pairs of courses. In general, the results with the internal measure $d_{I_{sd}}$ showed that courses of the same section typically have short distances. In the case of external distances this behavior is not clear, and external distances between courses of different sections are in some cases rather short.

The last comparison plots of the internal and external distances are given in Figure 3.16. These plots describe how the internal distances $d_{I_{sd}}$ and the different external distances $d_{E_{fr},P}$ between the ten different level

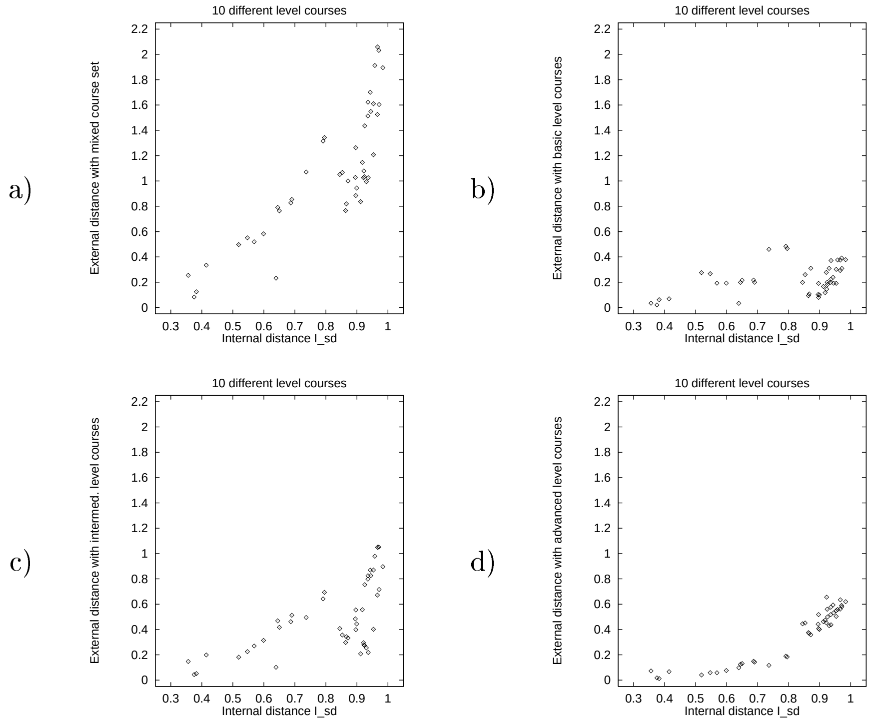


Figure 3.16: Comparison of internal distances $d_{I_{sd}}$ and external distances $d_{E_{fr},P}$ between ten different level courses with a) the mixed course probe set and its subsets of b) basic level, c) intermediate level, and d) advanced level courses.

courses in the course enrollment data correspond to each other. In this case the external distances seem to have a closer relationship to the internal distances than in our earlier test cases. However, these relationships are slightly different depending on the probe set used. By the internal distance measure $d_{I_{sd}}$ the most similar pair of courses is the pair *(Data Structures, Database Systems I)*. On the other hand, according to the external measures with the whole set of mixed probes and the subsets of the basic and the intermediate level probes, the most similar courses are *Computer Systems Organization* and *Information Systems*. This result is not very surprising because this pair of courses has the second shortest distance according to the measure $d_{I_{sd}}$. Moreover, with the advanced level probes, the most similar pair of courses is the pair *(Computer and Operating Systems, Database Systems I)*, which has the third shortest distance by the internal measure $d_{I_{sd}}$. In general, the external distances between courses from the same level are rather short, as are also their internal distances. Typically, the distances, either internal or external, are longest between the pairs of courses where one of the courses is a basic level course and the other an advanced level course. What the external distances between either the basic or the advanced level and the intermediate level courses are like, depends on the probe set used.

External distances with different probe sets

Our main idea in constructing the external similarity measure was to let the choice of probe attributes affect the distances. Therefore, given two sets P and Q of probe attributes 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. On the other hand, if the probe sets are associated with each other, the external distances obtained using them should also be related. Our experiments confirm both these conclusions.

The four plots in Figure 3.17 describe relationships between the external distances of 15 countries in the Reuters-21578 data when different probe sets are used. In these plots, the X axis describes the external distances with the probe set of food commodities, whereas the external distances represented on the Y axis differ. The points in the plots are very differently distributed indicating that different probe sets produce different similarity notions.

Three of the external measures used in this case agree on the most similar pair of countries. Namely, with the probe sets of food commodities, international organizations and mixed terms all regard the countries *Canada* and *USA* as the most similar pair of countries in the set of 15 countries considered. According to the other two external measures, the countries with the shortest distances are the pair *(Argentina, Mexico)* with the probe set of economic terms, and the pair *(Cuba, New Zealand)* with the probe set of energy terms. What is interesting is that while the external distance between *Cuba* and *New Zealand* with the energy term probes is zero, i.e., these countries are said to be completely similar, they are the most dissimilar pair of countries with the probe set of economic terms. This phenomenon can be explained as follows. The keyword *Cuba* does not occur often with any of the other keywords, neither the countries nor the various probe keywords. On the other hand, the keyword *New Zealand* does not co-occur with any of the chosen energy terms, but does co-occur with other types of probe keywords. Thus, in the case of energy terms the differences are small between the confidences $conf(Cuba \Rightarrow D_i)$ and

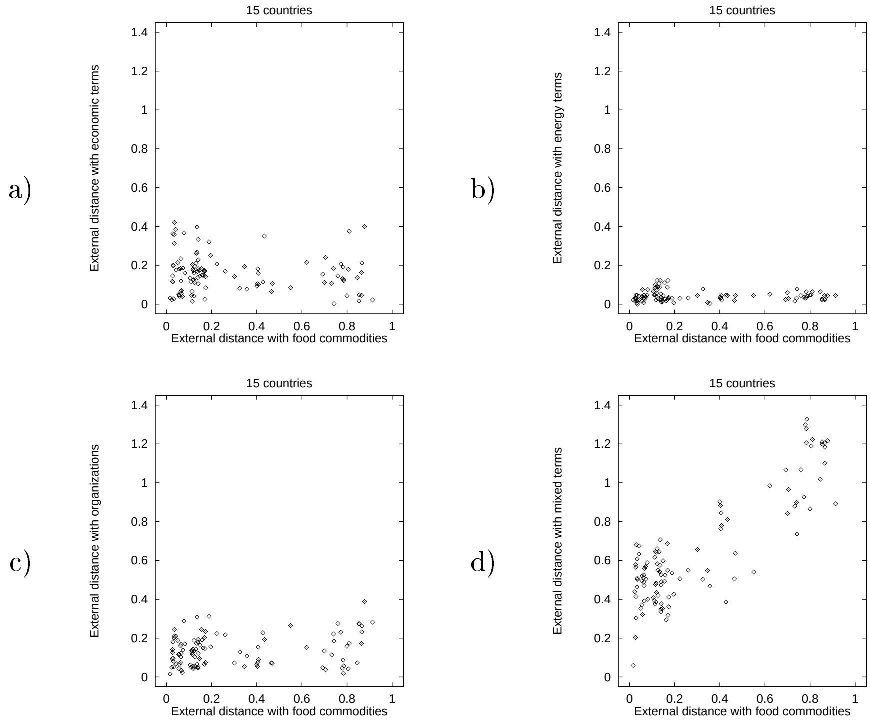


Figure 3.17: Comparison of external distances $d_{E_{fr},P}$ between 15 countries obtained using the food commodity probe set and the probe sets of a) economic terms, b) energy terms, c) organizations, and d) mixed terms.

$conf(New\ Zealand \Rightarrow D_i)$, where D_i is one of the energy terms, but in the case of the probe set of economic terms these differences are large.

That different probe sets produce different similarity notions is also confirmed by the orders of the external distances $d_{E_{fr},P}$ with different probe sets P. Consider, for example, keyword pairs *(Australia, Japan)* and *(Japan, USSR)* in the set of 15 countries. With the probe set of the international organizations, the external distances of both these pairs are 0.0416. However, with the probe set of mixed terms, their external distances are different, i.e., $d_{E_{fr},mixed\ terms}(Australia, Japan) = 0.6432$ and $d_{E_{fr},mixed\ terms}(Japan, USSR) = 1.1896$. This means that with respect to the mixed terms, *Australia* and *Japan* are more similar than *Japan* and *USSR*. On the other hand, using the probe set of energy terms, the external distances of these pairs are $d_{E_{fr},energy\ terms}(Australia, Japan) = 0.0720$ and $d_{E_{fr},energy\ terms}(Japan, USSR) = 0.0480$. That is, with respect to the energy terms, *Japan* and *USSR* are more similar than *Australia* and *Japan*.

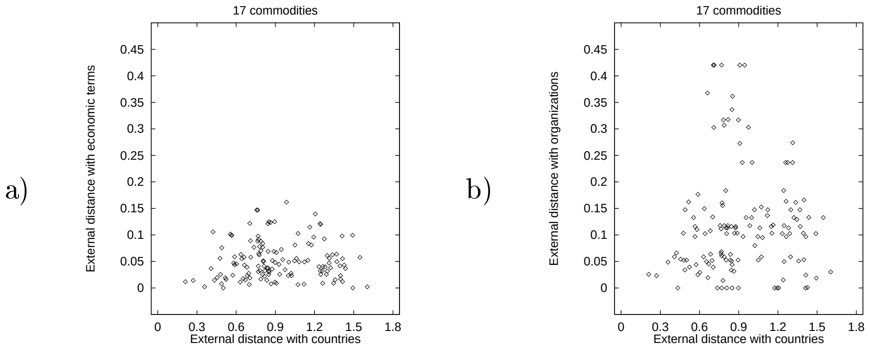


Figure 3.18: Comparison of external distances $d_{E_{fr},P}$ between 17 commodities obtained using the country probe set and the probe sets of a) economic terms, and b) organizations.

Therefore, the order of external distances is not the same with different probe sets.

Two comparison plots of external distances between the 17 commodities with different probe sets are shown in Figure 3.18. In these plots, the X axis describes external distances with the probe set of 15 countries and the Y axes external distances with the probe sets of economic terms (Figure 3.18a), and organizations (Figure 3.18b). The distributions of the points in these plots are different, and none of the three external measures agree on the most similar or the most dissimilar pair of commodities. The order of external distances between the 17 commodities is also different in every case. The external distances with the country probe set between the commodities *cocoa* and *coffee* is 0.6428, for example, and between the commodities *cocoa* and *copper* 1.2904. That is, *cocoa* and *coffee* are more similar commodities than *cocoa* and *copper* with respect to the 15 countries. On the other hand, with the organizations probe set, their external distances are the same, namely 0.1026, and thus, the two pairs of commodities are equally similar. With the third probe set, the set of economic terms, however, the order of the external distances of these pairs is changed. Namely, with the probe set of economic terms the external distances of these pairs are $d_{E_{fr},economic\ terms}(cocoa, coffee) = 0.0554$ and $d_{E_{fr},economic\ terms}(cocoa, copper) = 0.0256$. And this result is by no means exceptional in the set of 17 commodities.

In the same way, we compared the external distances between the two sets of courses in the course enrollment data using different probe sets. Three such comparison plots, where the X axis always describes the ex-

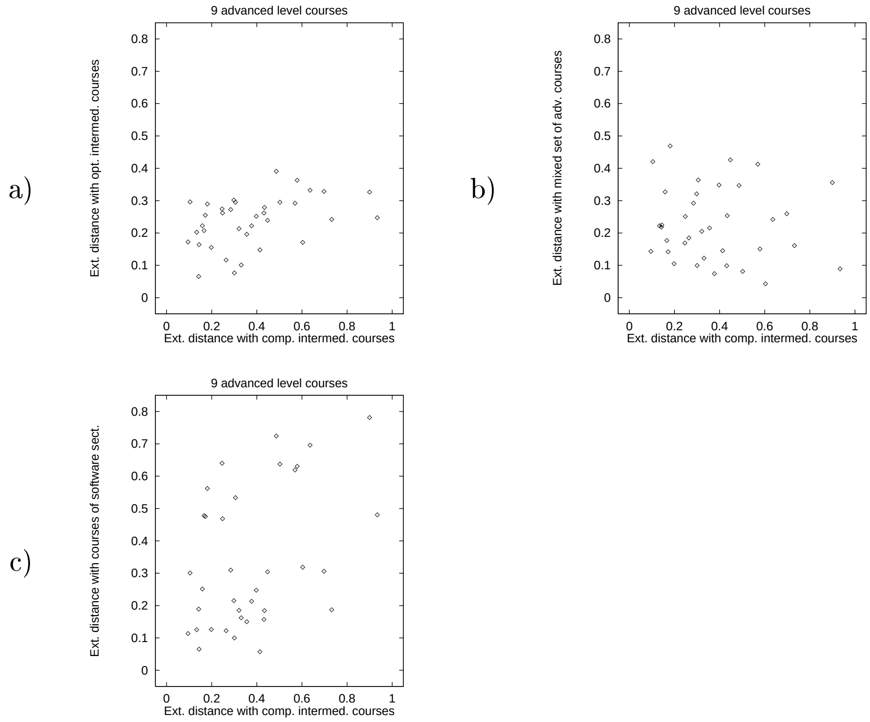


Figure 3.19: Comparison of external distances $d_{E_{fr},P}$ between nine advanced level courses obtained using the probe set of compulsory intermediate level courses and the probe sets of a) optional intermediate level courses, b) advanced level courses, and c) courses from the section of computer software.

ternal distances between the nine advanced level courses with the probe set of compulsory intermediate level courses, are given in Figure 3.19. In these plots the distributions of points vary a lot, and none of the four external measures states the same pair of courses as the most similar, or the most dissimilar pair. As the order of the external distances with different probe sets in general is also varying, these experiments further confirm our conclusion that by using different probe sets we can obtain very different similarity notions for binary attributes.

Finally, Figure 3.20 presents three comparison plots of external distances between ten courses from different levels in the course enrollment data. In these plots, the X axis always represents the external distances with the mixed course probe set, and the Y axes the external distances with the various subsets of it. Once again the distribution of the points

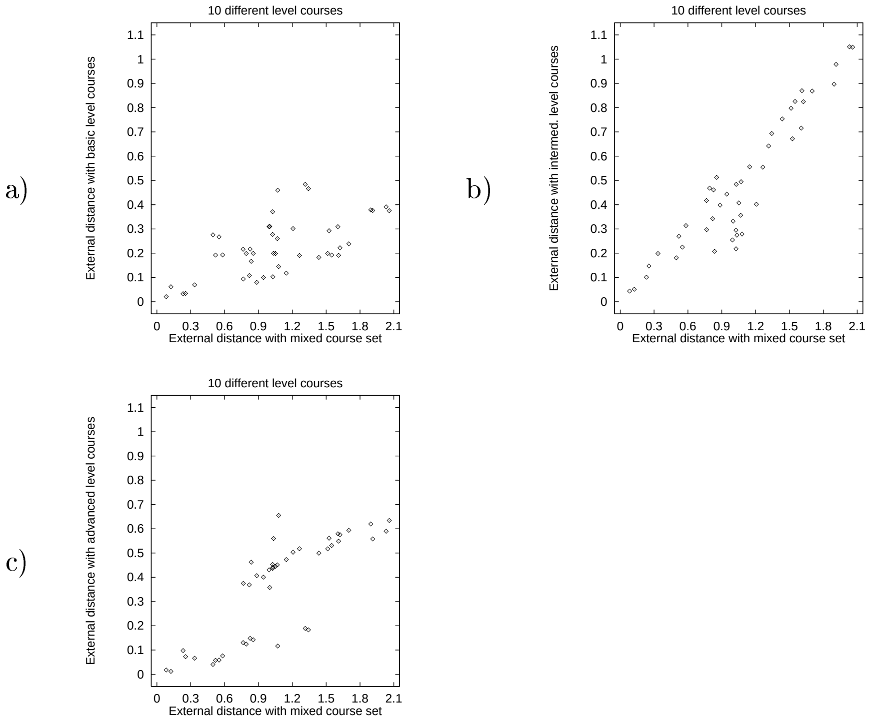


Figure 3.20: Comparison of external distances $d_{E_{fr},P}$ between ten different level courses obtained using the mixed course probe set and its subsets of a) basic, b) intermediate, and c) advanced level courses.

in the plots are different. All the four external measures regard a different pair of courses as the most dissimilar. However, three of the external measures agree on the most similar pair of courses. That is, the most similar pair of courses with the mixed probe set and its subsets of basic and intermediate level courses is the pair *(Computer Systems Organization, Information Systems)*. With the advanced level courses, on the other hand, the most similar courses are the courses *Computer and Operating Systems* and *Database Systems I*, but also the courses *Computer Systems Organization* and *Information Systems* are very similar.

Because three of the probe sets in this last case are subsets of the fourth probe set, i.e., the probe set of mixed courses, we can see that the external distances with different probe sets are now more closely related than in our earlier test cases. Especially, the relationship between the external distances with the mixed courses and the intermediate level courses is close to linear. These distinct relationships can be explained by the fact that the external distances with the whole mixed course probe set are actually sums of the external distances with the three other probe sets. Despite the fact that the external distances in this case are closely related, the order of distances still varies depending on the probe set used, and thus, these measures also define similarities between the ten chosen courses differently.

Attribute hierarchies

One of the reasons for studying attribute similarity is the need for building attribute hierarchies based purely on the data. In our experiments we built such attribute hierarchies based on the values of the different similarity measures presented in this chapter. In constructing these attribute hierarchies we used the standard *agglomerative hierarchical clustering* [And73, JD88, KR90], especially the single, complete and average linkage clustering (see Appendix A for details). In the following we give some examples of the attribute hierarchies constructed for two of the chosen sets of interesting attributes: the set of 15 countries and the set of nine advanced level courses.

Figure 3.21 presents four clustering trees for the set of 15 countries in the Reuters-21578 data set. All the clustering trees were produced with the single linkage method. The clustering tree based on the internal distance $d_{I_{sd}}$ (Figure 3.21a) reflects mainly the number of co-occurrences of the countries in the articles. The other three clustering trees are based on the external distances $d_{E_{fr},P}$ with the three probe sets P of food commodities (Figures 3.21b), international organizations (Figures 3.21c) and mixed terms (Figures 3.21d). All these trees weigh the co-occurrences of the countries with the given probe keywords, not with each other.

As the clustering trees in Figure 3.21 are based on different similarity measures, it is not surprising that they are different. Their differences depend on the distance measure used, and in the case of external measures the differences also reflect the different probe sets used. Despite the differences, these trees are all quite natural, and they correspond mainly to our views of the geopolitical relationships between the countries. For example, in Figure 3.21a the six G7 countries[2] together with *China* and *USSR* form a rather tight group. This is natural because articles about G7 countries usually name all member countries of the group. In the three other clustering trees we can also find many natural groups, for instance, a group of *Canada* and *USA*, or a group of Latin American countries. Note that the clustering tree in Figure 3.21b is quite a typical example of the chaining

[2]The seventh G7 country is Italy which did not belong to our test set of 15 countries.

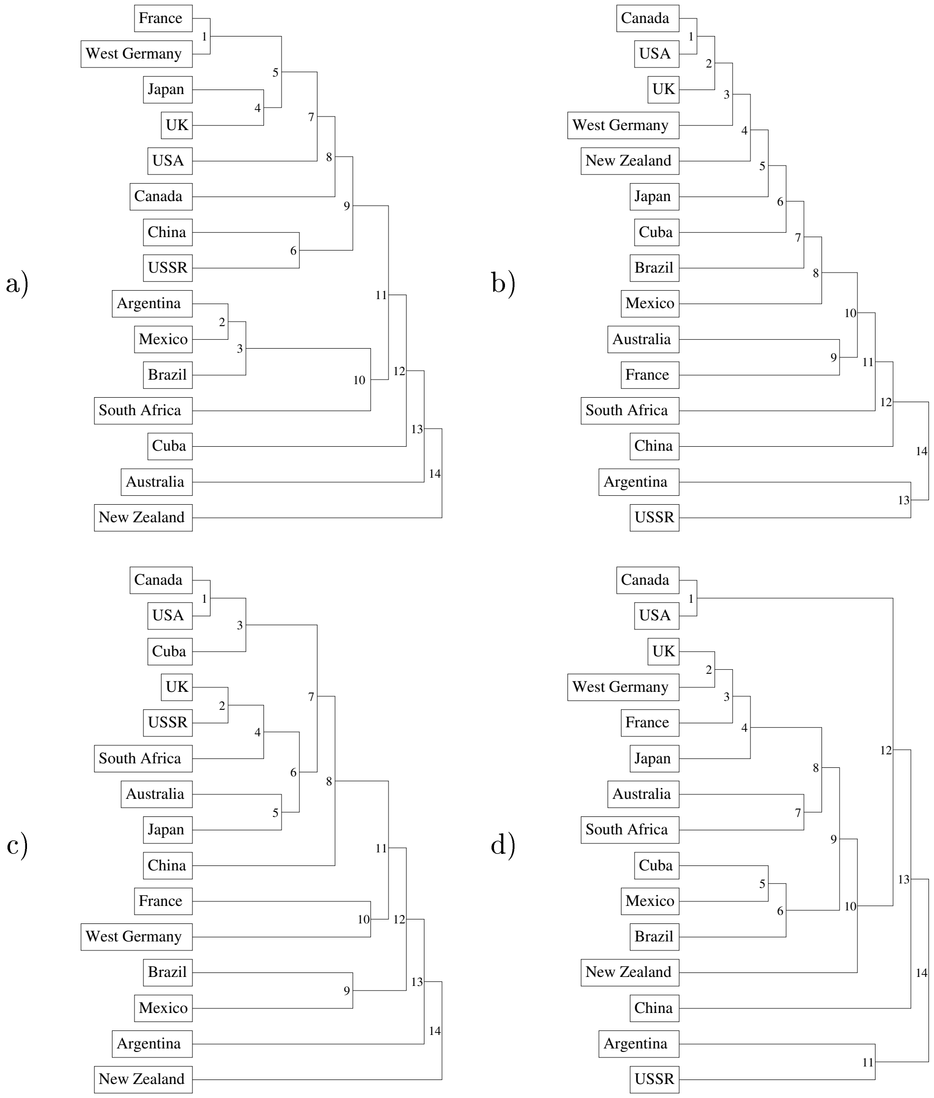


Figure 3.21: Clustering trees of 15 countries produced with the single linkage method by using a) the internal distances $d_{I_{sd}}$, and the external distances $d_{E_{fr},P}$ with the probe sets of b) food commodities, c) organizations, and d) mixed terms.

of clusters (see Appendix A), whereas the other three trees contain several clear groups of countries.

Four clustering trees for the nine advanced level courses in the course enrollment data are given in Figure 3.22. These trees are also produced using the single linkage method. Using the internal distance measure $d_{I_{sd}}$, we get the clustering tree in Figure 3.22a. In this tree the nine courses are clearly divided into three groups, corresponding to the sections at the Department of Computer Science. This clustering tree is very natural because many of the students mainly enroll in the courses of the same section.

Figure 3.22b presents the clustering tree of the courses based on the external distances $d_{E_{fr},P}$ when the optional intermediate level courses are used as probe attributes. In this tree there are two main clusters of courses. The first cluster consists of the courses from the section of computer software, and the second, the larger cluster, of the courses from the sections of the information systems and the general orientation in computer science. The clustering tree in Figure 3.22c, based on the external distances between the nine courses with the probe set of advanced level courses, also has two main clusters. These two clusters are, however, different from the previous case. The larger group of courses is now formed by the courses from the section of computer software and the section of information systems, whereas the smaller group contains just the courses of the general orientation in computer science. Using the external distances with the probe set of courses from the section of computer software, the clustering tree of the nine courses looks as in Figure 3.22d. In this case the courses *Distributed Operating Systems* and *Compilers* that are compulsory courses for the computer software section students form one cluster, whereas all the other courses are grouped together. These last three clustering trees once again confirm the idea that the probe sets describe the data from different points of view.

The clustering trees in Figures 3.21 and 3.22 show that when we use different distance measures, we can obtain clustering trees with very different structures, even though we use the same hierarchical clustering method. Moreover, if the distance values are computed using the same distance measure, but the clustering method is varied, we could get very different kinds of clustering trees for the chosen set of attributes. The hierarchical clustering methods differ in their way of defining the distance between two clusters. Therefore, it is only natural that the clusters that are merged in each phase of the clustering process do not have to be the same with every method, resulting in clustering trees with different structures.

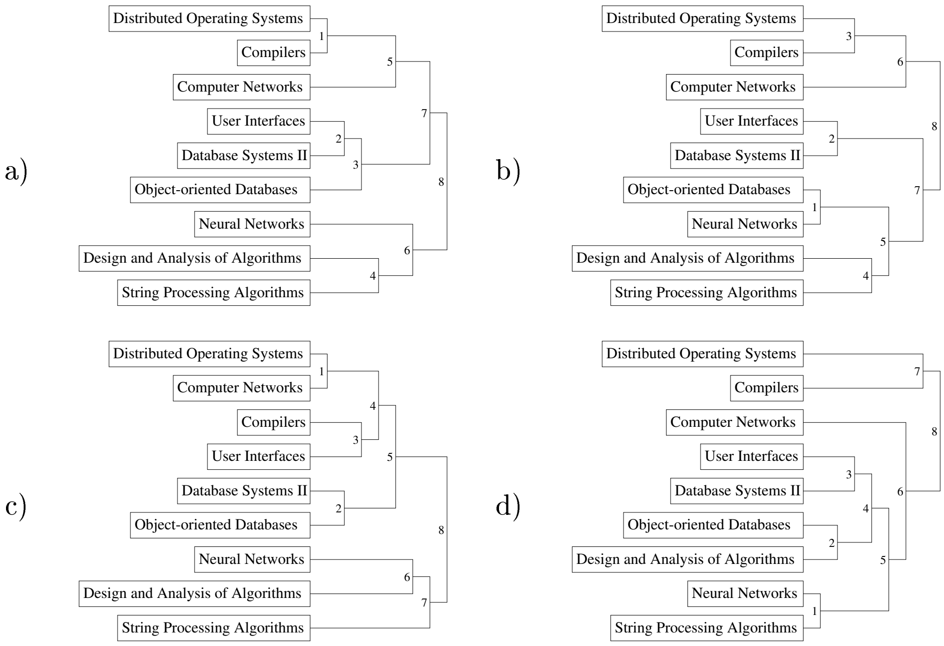


Figure 3.22: Clustering trees of nine advanced level courses produced with the single linkage method by using a) the internal distances $d_{I_{sd}}$, and the external distances $d_{E_{fr},P}$ with the probe sets of b) optional intermediate level courses, c) advanced level courses, and d) courses from the section of computer software.

Figure 3.23 presents a situation where the clustering trees produced with different hierarchical clustering methods differ from each other. In this figure there are three clustering trees of the nine advanced level courses formed by using the external distances between the courses with the probe set of compulsory intermediate level courses. During the first three phases of clustering, all the three methods proceed similarly, but after that different clusters are merged. The tree in Figure 3.23a is produced with the single linkage method. In this tree there are two clear clusters: a cluster of courses *Computer Networks* and *String Processing Algorithms*, and a cluster of the other seven courses. In Figure 3.23b there is a clustering tree of the nine courses produced with the complete linkage method. Now the courses are divided into three rather distinct groups. The first cluster is the same as in the tree formed using the single linkage method. The cluster of seven courses above is, however, divided in two clusters of four and three courses, respectively. The clustering tree in Figure 3.23c, which is produced using

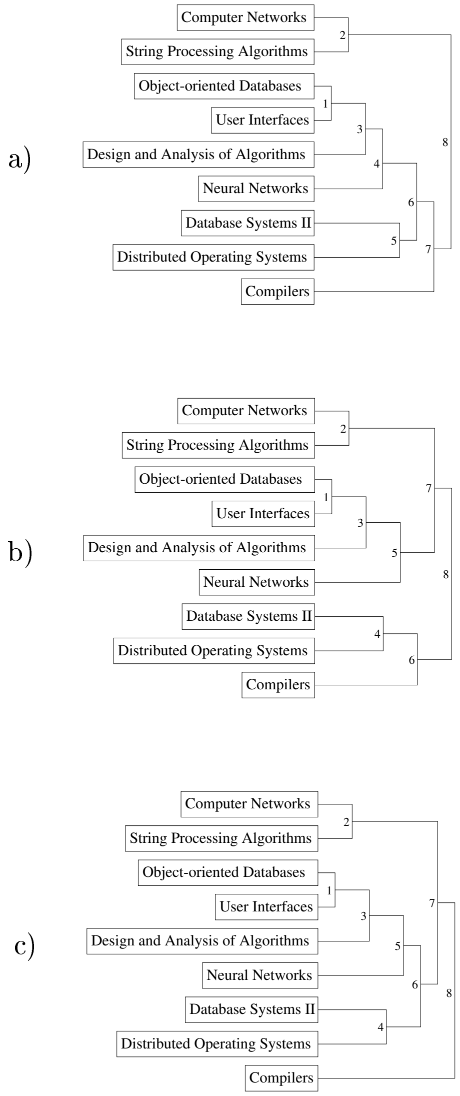


Figure 3.23: Clustering trees of nine advanced level courses produced with a) the single linkage, b) the complete linkage and c) the average linkage method using external distances $d_{E_{fr},P}$ with the probe set of compulsory intermediate level courses.

the average linkage method, also consists of three clear clusters. The first cluster is the same as in the two situations above. The second cluster consists of six courses, and what is somewhat surprising, the last one only of one course: the course *Compilers*. In this example case the differences in the clustering trees are rather small. In general, the clustering trees produced by the three different hierarchical linkage methods can either differ or, in some cases, they can be exactly the same.

3.6 Discussion

In this chapter we have presented several ways of measuring similarity between binary attributes. Using internal similarity measures we can get information on how two binary attributes co-occur in the rows of a relation. On the other hand, an external distance between two binary attributes weighs their co-occurrences with given probe attributes. Therefore, it is intuitively clear that these measures reflect different viewpoints on similarity between attributes.

In Section 3.5 we described results of our experiments with the distance measures using several real-life data sets. These results confirmed our expectations: internal and external distance measures truly describe different features of the data. For example, if two attributes do not co-occur at all, or only very seldom, their internal distance is inevitably long, i.e., these two attributes are internally dissimilar. However, with respect to some set of probe attributes these two attributes can be externally very similar, if they co-occur with those probe attributes in the same way. Yet, if two attributes are externally similar with respect to one probe set, they can be very dissimilar, when another set of probe attributes is considered. This is also expected: the probe set defines the point of view from which similarity between attributes is judged. There is just one exception from these rules above. Namely, if two attributes are very similar according to an internal distance measure, they cannot be dissimilar according to any external distance measure. In those cases they inevitably co-occur in the same way with most of the possible probe attributes.

In a typical situation, there is no single notion of similarity between binary attributes, and neither can we give any general rules about which measure to choose. Therefore, the choice of the distance measure between binary attributes mainly depends on for what kind of similarity we are searching. In some cases, the application area may also influence this choice.

Chapter 4

Similarity between event sequences

In this chapter we move to discuss another important type of data considered in data mining, i.e., sequences of events. Especially we consider how we could define similarity between sequences of events. Our approach to this problem is based on an idea that similarity between two event sequences should reflect the amount of work needed to transform one sequence into another. We formalize this idea as an *edit distance* between event sequences, and show that using such a measure with different parameter values we can achieve very different notions of event sequence similarity. We also show how these similarities between sequences can be used, for example, to build hierarchies of sequences.

We start by describing properties of event sequences in Section 4.1. In Section 4.2 we introduce different measures for similarity between event sequences. After that, in Section 4.3 we give algorithms for computing these similarities. Experimental results on event sequence similarity are presented in Section 4.4, and the meaning of these results is discussed in Section 4.5. A part of the material in this chapter is based on [MR97].

4.1 Event sequences

Often, when we use data mining techniques, we study unordered data sets. Still, in many important application areas the data has a clear sequential structure. In user interface studies, in World Wide Web (WWW) page request monitoring, or in telecommunication network monitoring, for example, a great deal 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 an $(m+1)$-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 ranged in an ascending order by their occurrence times. □

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

Example 4.1 Telecommunication networks produce 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 also has a type and an occurrence time associated with it. An example of a real alarm is

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

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

Example 4.2 Page requests made in the WWW 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

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

where the first attribute is the requested page, and the second the requesting host. The next two attributes of the request describe the success status of the request, and the last attribute is the occurrence time 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, the alarm text can be considered redundant if it can be determined from the alarm type. On the other hand, one example of what can be regarded as irrelevant information is the physical location of the network element sending the alarm. If such redundant and irrelevant attributes are disregarded, only few of the event attributes are really interesting when studying the similarity between event sequences. In the following, we therefore consider a simpler model of events where each event has only a type and an occurrence time.

Definition 4.2 Let the set $\mathcal{E}$ be a set of *event types*. Now 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. An event sequence $\mathcal{S}$ is then 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 $e_i \in \mathcal{E}$ for all $i = 1, \ldots, n$, and $t_i \leq t_{i+1}$ for all $i = 1, \ldots, n-1$. The length of the sequence $\mathcal{S}$ is denoted by $|\mathcal{S}| = n$. A sequence that only consists of event types in temporal order, i.e.,

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

where each $e_i \in \mathcal{E}$ for all $i = 1, \ldots, n$, is called an *event type sequence*. An empty event sequence, or an empty event type sequence is denoted by $\langle\rangle$. □

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

$$\mathcal{S} = \langle\, (A, 30),\ (D, 31),\ (F, 32),\ (E, 33),\ \ldots,\ (E, 66)\, \rangle.$$

An event type sequence corresponding to this event sequence is

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

Both the event sequence $\mathcal{S}$ and the event type sequence S contain 25 events. If, however, there were events whose occurrence times were less than 30 or more than 66, and they were taken into account, the sequences would have been longer. □

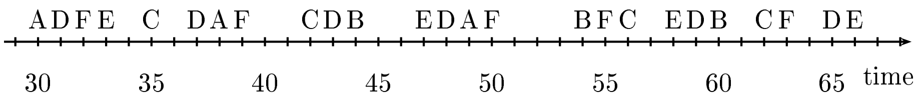


Figure 4.1: An event sequence on the time axis.

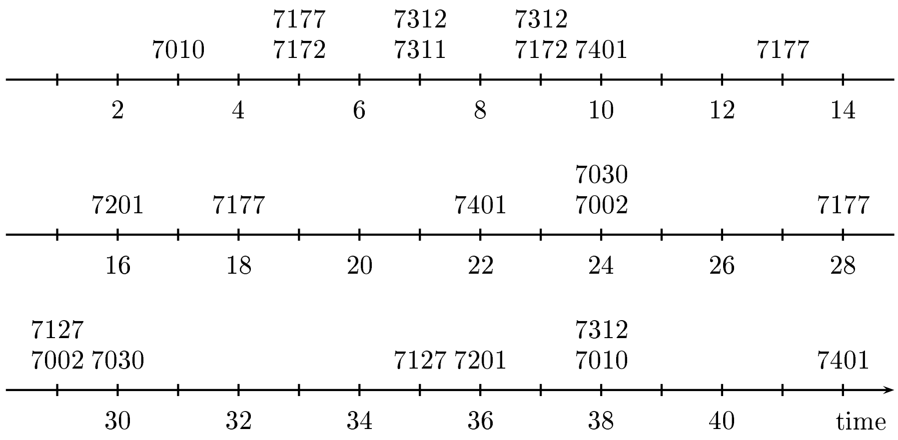


Figure 4.2: An example alarm sequence on the time axis.

Example 4.4 An example of an event sequence in telecommunication data is an alarm sequence $\mathcal{S}_{alarm}$ given in Figure 4.2. This sequence consists of 23 alarms from the set

$$\mathcal{E} = \{7002, 7010, 7030, 7127, 7172, 7177, 7201, 7311, 7312, 7401\}$$

of alarm types which is only a small alarm type set. Typically, there are hundreds, or even thousands of different types of alarms in alarm sequences.

One problem with analyzing alarm data is related to the occurrence times of the alarms in the network. In the alarm sequence $\mathcal{S}_{alarm}$ in Figure 4.2 there are several cases where two alarms have the same occurrence time. For instance, alarms of types 7172 and 7177 occur at time 5. The problem here is that we cannot know for certain which is the real order of these alarms. When analyzing alarm sequences we should actually take into account all the possible orders of alarms that have the same occurrence time. 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 may have occurred exactly at the same time, but still have different occurrence times in the given sequence, or an alarm can be assigned an occurrence time which is different from its real occurrence time.

Despite these known problems with occurrence times, we assume that alarms have occurred strictly in the order in which they are written into the alarm log, and leave handling of the problems with occurrence times of alarms for future study. Hence, the sequence in Figure 4.2 is considered to be the event sequence

$$\begin{aligned}\mathcal{S}_{alarm} &= \langle\,(7010,3),(7172,5),(7177,5),(7311,7),\\ &\quad (7312,7),\ldots,(7010,38),(7312,38),(7410,42)\,\rangle,\end{aligned}$$

i.e., when two events in the sequence have the same occurrence time, the alarm in the upper position in the figure was written later in the alarm log. Then the sequence

$$S_{alarm} = \langle\,7010,7172,7177,7311,7312,\ldots,7010,7312,7410\,\rangle$$

is the event type sequence corresponding to the alarm sequence $\mathcal{S}_{alarm}$. □

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

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

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

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

In this chapter, we use letters from the beginning of the alphabet, like A, B and C, to denote event types. A set of all possible event types is denoted by $\mathcal{E}$, event sequences by a calligraphic letter $\mathcal{S}$, and event type sequences by a letter S.

Real-life event sequences are often 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 an event subsequence.

Definition 4.3 Let $\mathcal{S}$ be an event sequence and S an event type sequence over a set $\mathcal{E}$ of event types. A boolean expression θ on event types and/or occurrence times of events is called a *selection condition* on events of a sequence. An *event subsequence* of the sequence $\mathcal{S}$ is an event sequence that satisfies θ, i.e.,

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

An *event type subsequence* can either be an event type sequence that satisfies θ in the event sequence $\mathcal{S}$, i.e.,

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

or an event type sequence that satisfies θ in the event type sequence S, i.e.,

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

□

The definitions of an event subsequence $\mathcal{S}(\theta)$ and an event type subsequence $S(\theta)$ resemble the definition of a subrelation r_θ in Section 3.1. The form of a selection condition θ in Definition 4.3 was, however, left quite open. The condition θ can contain restrictions on event types, occurrence times of the events, 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 boolean operators.

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

Note that a subsequence does not have to be a continuous 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 select a subsequence, for example,

$$\mathcal{S}(e_i = A) = \langle (A, 30), (A, 38), (A, 49) \rangle$$

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

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 7177 consists of four events, i.e.,

$$\mathcal{S}(e_i = 7177) = \langle (7177, 5), (7177, 13), (7177, 18), (7177, 28) \rangle.$$

Assume then that we are interested in what happens, for instance, at most five seconds before an event of type 7401 occurs, i.e., we want to find subsequences $\mathcal{S}(\,\exists\,(e_j, t_j)$ so that $e_j = 7401 \,\wedge\, t_j - 5 \leq t_i \leq t_j \,\wedge\, i < j\,)$. In the example sequence there are three events of type 7401, at times 10, 22 and 42. Therefore, we find three subsequences

$$\begin{aligned}
&\mathcal{S}_1 = \langle\, (7172, 5), (7177, 5), (7311, 7), (7312, 7), (7172, 8), (7312, 8) \,\rangle, \\
&\mathcal{S}_2 = \langle\, (7177, 18) \,\rangle, \text{ and} \\
&\mathcal{S}_3 = \langle\, (7010, 38), (7312, 38) \,\rangle.
\end{aligned}$$

On the other hand, a subsequence

$$\mathcal{S}_{alarm}(10 \leq t_i \leq 20) = \langle\, (7401, 10), (7177, 13), (7201, 16), (7177, 18) \rangle$$

is an alarm subsequence which consists of alarms with occurrence times between 10 and 20.

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 7100, i.e., a subsequence $\mathcal{S}(7000 \leq e_i \leq 7100)$. The resulting subsequence consists of seven events and is

$$\begin{aligned}
\mathcal{S}(7000 \leq e_i \leq 7100) \quad = \quad & \langle\, (7010, 3), (7002, 24), (7030, 24), (7002, 29), \\
& (7030, 30), (7010, 38) \,\rangle.
\end{aligned}$$

□

In recent years, interest in knowledge discovery from event sequences has been quite extensive; see, for example, [AS95, GRS99, GWS98, HKM+96, Lai93, MKL95, MT96, MTV95, MTV97, OC96, WH98, Zha99]. The problem of similarity between sequences has also been considered in many areas such as text databases, genetics, biomedical measurements, telecommunication network measurements, economic databases, and scientific databases. This research, however, often concentrates on sequences of numerical values, not on sequences of events. Especially time series and similarity queries on them have been widely studied; see, for example, [AFS93, ALSS95, APWZ95, DGM97, FRM93, GK95, RM97, SHJM96]. Some of these articles consider similarity between long sequences, and some of them just finding similar subsequences.

In this thesis we consider how similarity between two event sequences can be defined, but also how to define similarity between two event type sequences. That is, we study similarities between two sequences with or without occurrence times. In both of these cases, we define similarity between two sequences by using a complementary notion of distance.

Definition 4.4 Given a set of event types $\mathcal{E}$ and a class of all possible sequences $\mathbf{S}$ over $\mathcal{E}$, a *distance* measure d between sequences is defined as $d : \mathbf{S} \times \mathbf{S} \to \mathrm{I\!R}$. Then, the distance between two event sequences $\mathcal{S}_1$ and $\mathcal{S}_2$ in the set $\mathbf{S}$ is denoted by $d(\mathcal{S}_1, \mathcal{S}_2)$. Similarly, given two event type sequences S_1 and S_2 belonging to the set $\mathbf{S}$, the distance between them is denoted by $d(S_1, S_2)$. □

There are several ways of defining the distance between two event sequences, or between two event type sequences. The intuitive idea behind our distance measure is that it should reflect the amount of work needed to transform one sequence into another. In Section 4.2 we show how this idea is formalized as *edit distance*, which is a common and simple formalization of a distance between strings or sequences. Edit distance is a widely used distance measure 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, YÖ96] and behavioral sequences [LB97].

Sometimes we are interested in just computing similarity, or distance between two sequences. We may, however, also be interested in a set of sequences, and want to find out how similar to each other all these sequences are. This leads us to the following definition.

Definition 4.5 Let $\mathcal{E}$ be a set of event types and $\mathbf{S}$ a set of all sequences over $\mathcal{E}$. Assume then that we have a set of event sequences, or event type sequences that belong to the set $\mathbf{S}$, or are subsequences of the sequences in the set $\mathbf{S}$, and that fulfill some selection criterion on event types and/or occurrence times of events. If we want to compute the pairwise similarities between the sequences in this set, it is called the *set of interesting sequences*, and it is denoted by $\mathcal{SI}$. The size of such a set, i.e., the number of sequences in such a set, is denoted by $|\mathcal{SI}|$. □

Example 4.8 In telecommunication alarm data, a set $\mathcal{SI}$ of interesting alarm sequences could be, for example, all sequences of events preceding occurrences of a certain alarm type, like the alarm type 7010, within some time period. Another set $\mathcal{SI}$ of interesting sequences might be all sequences of alarms sent by a certain network element in weekdays between 7 and 10 o'clock in the evening. □

Example 4.9 A set $\mathcal{SI}$ of interesting sequences in WWW might be a set of sequences of page requests made by a user during separate sessions. Another set of interesting page request sequences could consist of sequences preceding a request to a certain WWW page, for example the home page of the Data Mining research group at the University of Helsinki, within a given time period. □

4.2 Similarity measures

As stated in the previous section, our intuitive idea of similarity, or distance between sequences of events is based on how much work is needed in transforming one sequence of events into another. Such a distance can be formalized as edit distance, for example. In this section we show how to define edit distance both between two event sequences and between two event type sequences.

4.2.1 Edit operations

A transformation of one sequence of events into another is made by a sequence of edit operations on events of the sequence.

Definition 4.6 Let $\mathcal{O}$ be a set of *edit operations* allowed in a transformation between sequences. A transformation between two sequences of events can be presented by giving a series of needed edit operations, called an *operation sequence.* An operation sequence of k edit operations is denoted by $O = \langle o_1, o_2, \ldots, o_k \rangle$, where each $o_i \in \mathcal{O}$ for all $i = 1, \ldots, k$. □

The set of allowed edit operations depends on the application area, our purpose, and the type of the sequences. In traditional text string matching and biosequence comparison, a standard edit operation set consists of an insertion, a deletion, and a substitution of characters. Whatever edit operations are chosen, a non-operation called *match* is always used in computing the edit distance.

In transforming one event sequence into another, we use three parameterized edit operations:

1. $\texttt{Ins}\,(e_i, t_i)$ that inserts an event of type e to a time t_i, i.e., adds an event (e_i, t_i) to a sequence,

2. $\texttt{Del}\,(e_i, t_i)$ that deletes an event of type e from a time t_i, i.e., deletes an event (e_i, t_i) from a sequence, and
3. $\texttt{Move}\,(e_i, t_i, t_i^{'})$ that changes the occurrence time of an event (e_i, t_i) from time t_i to time $t_i^{'}$.

All these three edit operations also take into account the occurrence times of events. They are very natural and useful when computing the edit distance between two event sequences. In the following, we denote the set $\{\texttt{Ins}, \texttt{Del}, \texttt{Move}\}$ of edit operations by $\mathcal{O}_{\mathcal{S}}$. Note that we do not allow transpositions, or swaps of adjacent events, not even in the case when the occurrence times of events are changed.

When considering two event type sequences, however, we use only two parameterized edit operations: an operation $\texttt{Ins}\,(e_i)$ that inserts an event of type e into a sequence as its i'th event, and an operation $\texttt{Del}\,(e_i)$ that deletes an event of type e from a sequence, when this event is the i'th event of the sequence from which the transformation began. These two operations were chosen, because they are very natural, and they can easily be applied with every kind of event type sequences. This set $\{\texttt{Ins}, \texttt{Del}\}$ of edit operations we denote by $\mathcal{O}_S$.

Example 4.10 Consider an event type set $\mathcal{E} = \{A, B, C, D, E\}$, two event sequences $\mathcal{S}_1 = \langle (A,1), (B,3), (A,4), (C,6), (B,9), (D,11) \rangle$ and $\mathcal{S}_2 = \langle (A,2), (B,5), (C,8), (C,12), (A,13), (D,16) \rangle$ over $\mathcal{E}$ (also presented in Figure 4.3), and the set $\mathcal{O}_{\mathcal{S}}$ of edit operations. An operation sequence

$$\begin{aligned} O \;=\; \langle\, & \texttt{Move}\,(A,1,2),\ \texttt{Move}\,(B,3,5),\ \texttt{Del}\,(A,4), \\ & \texttt{Move}\,(C,6,8),\ \texttt{Del}\,(B,9),\ \texttt{Ins}\,(C,12), \\ & \texttt{Ins}\,(A,13),\ \texttt{Move}\,(D,11,16)\,\rangle \end{aligned}$$

is one of the possible operation sequences transforming the sequence $\mathcal{S}_1$ into the sequence $\mathcal{S}_2$.

Then consider two event type sequences $S_1 = \langle A, B, A, C, B, D \rangle$ and $S_2 = \langle A, B, C, C, A, D \rangle$ corresponding to the event sequences above, and the set $\mathcal{O}_S$ of edit operations. Now an operation sequence

$$O = \langle\, \texttt{Del}\,(A_3),\ \texttt{Ins}\,(C_4),\ \texttt{Del}\,(B_5),\ \texttt{Ins}\,(A_5)\,\rangle$$

is one of the operation sequences transforming the event type sequence S_1 into the event type sequence S_2. $\square$

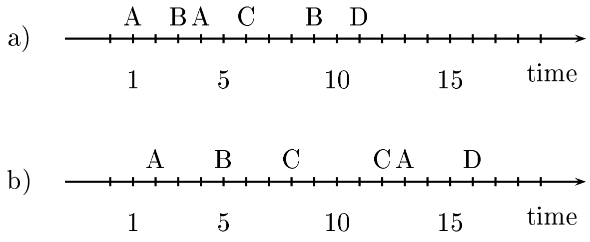


Figure 4.3: The event sequences a) $\mathcal{S}_1$ and b) $\mathcal{S}_2$ of Example 4.10 on the time axis.

4.2.2 Costs of operations

In traditional text matching and comparison of biosequences, the edit distance between two strings or sequences is often defined as the minimum number of edit operations needed to transform a string (sequence) into another. This type of an edit distance is also referred to as the *Levenshtein distance* [CR94, Ste94, Gus97]. If we use such an approach for defining the distance between sequences of events, then our intuition says that two sequences are similar, if many of the events in them match and the operation sequence O contains only few insertions and deletions. In the case of event sequences such an operation sequence should also contain only few moves, or at least, the moves should be short.

We want, however, the measure of the distance between sequences of events to be more general than just based on the number of the needed edit operations. This leads us to the definition of edit operation costs.

Definition 4.7 Let $\mathcal{O}$ be a set of edit operations allowed. We associate with every edit operation $o \in \mathcal{O}$ a *cost*[1] which we denote by $\mathsf{c}(o)$. □

There are several ways of defining the costs of edit operations. If each operation in the set $\mathcal{O}$ has a predefined constant cost, edit operations are said to have *operation-weight costs*. On the other hand, if the cost of an operation depends on the type of the event considered, we say that edit operations have *alphabet-weighted costs*.

We start by considering costs of `Ins`- and `Del`-operations. The simplest way of assigning costs for these operations is to use constant *unit costs*. In

[1]Also the terms *weight* and *score* of an edit operation are widely used in literature; the first one in the computer science literature, and the second one in the biological literature [Gus97].

the case of event type sequences this means that the cost of inserting an event e_i to a sequence is $\mathsf{c}\,(\mathtt{Ins}\,(e_i)) = 1$, and the cost of deleting an event e_i from a sequence is $\mathsf{c}\,(\mathtt{Del}\,(e_i)) = 1$, for every event type in $\mathcal{E}$. Similarly, in the case of event sequences, the unit cost of inserting an event (e_i, t_i) to a sequence is $\mathsf{c}\,(\mathtt{Ins}\,(e_i, t_i)) = 1$, and the cost of deleting an event (e_i, t_i) from a sequence is $\mathsf{c}\,(\mathtt{Del}\,(e_i, t_i)) = 1$, for every event type in $\mathcal{E}$ and for every occurrence time $t_i \in \mathbb{R}$.

Because some types of events occur more often in sequences than others, it might be more natural, if the cost of an edit operation was dependent on the type of the event. We may, for example, want the cost of adding (deleting) a rare event to (from) a sequence to be higher than the cost of adding (deleting) a common event. For event type sequences, we define an alphabet-weighted cost of an Ins-operation as

$$\mathsf{c}\,(\mathtt{Ins}\,(e_i)) = w(e),$$

where $w(e)$ is a constant proportional to $occ(e)^{-1}$, when $occ(e)$ is the number of occurrences of an event type e in a long reference sequence from the same application area. An alphabet-weighted cost of deleting an event from an event type sequence is, in turn, defined to be the same as the cost of an Ins-operation, i.e., $\mathsf{c}\,(\mathtt{Del}\,(e_i)) = w(e)$. Similarly, we can define alphabet-weighted costs of inserting or deleting an event from an event sequence to be $\mathsf{c}\,(\mathtt{Ins}\,(e_i, t_i)) = w(e)$ and $\mathsf{c}\,(\mathtt{Del}\,(e_i, t_i)) = w(e)$. Note that here we have defined the costs of Ins- and Del-operations to be exactly the same, but in general they could also differ from each other.

A more difficult problem than defining the costs of inserting and deleting events is to define the cost of moving an event in time when transformations between event sequences are considered. We define this cost as

$$\mathsf{c}\,(\mathtt{Move}\,(e_i, t_i, t_i^{'})) = V \cdot |\, t_i - t_i^{'}|,$$

where V is a constant and $|\, t_i - t_i^{'}|$ is the length of the move. With this definition a short move has a lower cost than a long move. For two event sequences to be considered similar, this definition assumes that the occurrence times of events have approximately the same magnitude in both compared sequences. This means that sequences such as $\langle (A, 1), (B, 2), (C, 5) \rangle$ and $\langle (A, 101), (B, 102), (C, 105) \rangle$ are considered to be very far from each other, even though the types of their events and the distances between the occurrence times of their events in time match exactly.

In some applications it may be useful, or even necessary to limit the length of moves of the events explicitly. A bound W for the maximum length of moves, a *window width*, can be a predefined value in given time units. It can also be somehow dependent on the occurrence times in the sequences: it can be the maximum of the lengths of the time periods between the first and the last event in the sequences, for example, or the length of the longest or the shortest of the sequences. With this bound, the cost of moving an event is always $\mathsf{c}\,(\mathtt{Move}\,(e_i, t_i, t_i')) \leq V \cdot W$.

The parameter V used in defining the cost of a Move-operation also has some logical restrictions. In the case of unit costs for inserting and deleting an event, we should have $V \leq 2$. Namely, if $V > 2$, then moving an event is never cost-effective: instead, one can always first delete and then insert an event of the type e. Note that in this case the cost of a Move-operation does not depend on the event types. For similar reasons, when we use alphabet-weighted costs for insertions and deletions of events, we should 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 for every event type is $2 \cdot \mathsf{min\_w}$ where $\mathsf{min\_w} = \min\{w(e)\}$. In this case, the cost of moving an event does not directly depend on the type of the event either. However, the length of a cost-effective move is nearly always dependent on the type of the event. To name an example, if $V = 2 \cdot \mathsf{min\_w}$, then for an event of the type e which has the minimum weight $\mathsf{min\_w}$, the length of the longest cost-effective move is one time unit. The length of the longest cost-effective move for an event of any other type is determined by an equation

$$|t_i - t_i'| \leq \frac{w(e)}{\mathsf{min\_w}},$$

where $w(e)$ is a constant inversely proportional to the number of occurrences of the event type e. The absolutely longest moves are cost-effective for events of the type e with the maximal value of $w(e)$ among all the event types in $\mathcal{E}$. There are, however, values of the parameter V with which the length of a cost-effective move is never dependent on the type of the event. If we use a parameter value $V = \frac{2 \cdot \mathsf{min\_w}}{W}$, for example, then all moves of W time units or less would be cost-effective regardless of the type of the event considered.

As noted before, transforming one sequence into another usually demands several edit operations. A total cost of such an operation sequence can be defined as follows.

Definition 4.8 Let $\mathcal{O}$ be a set of edit operations, and $\mathsf{c}\,(o)$ a cost for an operation $o \in \mathcal{O}$. A cost of an operation sequence $O = \langle o_1, o_2, \ldots, o_k \rangle$ is the sum of the costs of all individual operations o_i, i.e.,

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

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

4.2.3 Edit distance between sequences

We now give a formal definition of the edit distance between two event sequences, or between two event type sequences.

Definition 4.9 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 between these event sequences $\mathcal{S}_1$ and $\mathcal{S}_2$ is defined as

$$d_{\mathcal{S}}(\mathcal{S}_1, \mathcal{S}_2) = \min \{ \ \mathsf{c}\,(O_j) \mid O_j \text{ is an operation sequence transforming sequence } \mathcal{S}_1 \text{ into sequence } \mathcal{S}_2 \ \},$$

i.e., as the cost of an optimal operation sequence[2] $\hat{O}$ transforming the sequence $\mathcal{S}_1$ into the sequence $\mathcal{S}_2$. Similarly, the edit distance d_S between two event type sequences $S_1 = \langle e_1, e_2, e_3, \ldots, e_m \rangle$ and $S_2 = \langle f_1, f_2, f_3, \ldots, f_n \rangle$ is defined as the cost of the optimal operation sequence $\hat{O}$ transforming the sequence S_1 into the sequence S_2. □

If the edit distance between two event sequences, or event type sequences, is determined using Definitions 4.8 and 4.9, we are actually considering a *weighted edit distance*. More specifically, if edit operations have operation-weighted costs, we should talk about *operation-weight edit distance*, and if they have alphabet-weighted operations costs, we should use a term *alphabet-weight edit distance* [Gus97].

If the edit distance between sequences of events had been defined as the minimum number of operations needed to transform a sequence into another, then our distance measure would have been a metric. However, when we define the edit distance between sequences as the cost of the optimal operation sequence making such a transformation, we cannot be sure that the resulting measure is a metric. Especially, with alphabet-weighted operation costs the triangle inequality might not be satisfied, and, therefore, the measure obtained is not necessarily a metric [Ste94]. This does not, however, prevent the use of weighted edit distance as a measure of distance between sequences.

Example 4.11 Let $\mathcal{E} = \{A, B, C, D, E\}$ be the set of event types, and $\mathcal{O}_S$ the set of edit operations. Using unit operation costs, an event type

[2]Note that there can be several operation sequences that have the minimum cost, and thus, there can also be many optimal operation sequences.

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

Table 4.1: The numbers of occurrences of different event types occurring in a hypothetical reference sequence and the alphabet-weighted operation costs of inserting and deleting events of these types.

sequence $S_1 = \langle A, B, A, C, B, D \rangle$ can be transformed into an event type sequence $S_2 = \langle A, B, C, C, A, D \rangle$, for instance, with three operation sequences O_1, O_2 and O_3. That is, with operation sequences

$$O_1 = \langle\, \mathtt{Del}\,(A_3),\ \mathtt{Ins}\,(C_4),\ \mathtt{Del}\,(B_5),\ \mathtt{Ins}\,(A_5)\,\rangle,$$

$$O_2 = \langle\, \mathtt{Ins}\,(C_3),\ \mathtt{Del}\,(A_3),\ \mathtt{Del}\,(B_5),\ \mathtt{Ins}\,(A_5)\,\rangle,$$

and

$$O_3 = \langle\, \mathtt{Ins}\,(C_3),\ \mathtt{Ins}\,(C_4),\ \mathtt{Del}\,(C_4),\ \mathtt{Del}\,(B_5)\,\rangle.$$

The costs of each of these operation sequences are $\mathtt{c}\,(O_1) = \mathtt{c}\,(O_2) = \mathtt{c}\,(O_3) = 4$. They are all optimal operation sequences, and thus, the weighted edit distance between the event type sequences S_1 and S_2 is 4.

Suppose then that we have a reference sequence from which we get the numbers of occurrences of different event types in the set $\mathcal{E}$. Let Table 4.1 present the numbers of these occurrences and the alphabet-weighted costs of insertions and deletions of these event types, when $\mathtt{c}\,(\mathtt{Ins}\,(e_i)) = \mathtt{c}\,(\mathtt{Del}\,(e_i)) = w(e) = occ(e)^{-1}$. Now the weighted edit distance between the event type sequences S_1 and S_2 is

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

This distance is the cost of the operation sequences O_1 and O_2, which are both optimal operation sequences. However, the operation sequence O_3 is no longer optimal, because its cost is 0.17, which is higher than the minimum cost 0.09. □

Example 4.12 Let $\mathcal{E} = \{A, B, C, D, E\}$ be the set of event types, and $\mathcal{S}_1$ and $\mathcal{S}_2$ the two event sequences in Figure 4.3. Assume that we have a window width $W = 20$ and the set $\mathcal{O}_\mathcal{S}$ of edit operations.

If the edit operations have the unit costs, the value of the parameter V should be less than 2. For example, with value $V = 0.5$, the cost of moving an event is $\mathtt{c}\,(\mathtt{Move}\,(e_i, t_i, t_i^{'})) = 0.5 \cdot |\, t_i - t_i^{'}|$, and the cost of the maximum length move (20 time units) is, therefore, $0.5 \cdot 20 = 10$. With these operation costs, the optimal operation sequence $\hat{O}$ for the sequences $\mathcal{S}_1$ and $\mathcal{S}_2$ is

$$\begin{aligned}\hat{O} \;=\; & \langle\, \mathtt{Move}\,(A,1,2),\ \mathtt{Move}\,(B,3,5),\ \mathtt{Del}\,(A,4),\\ & \mathtt{Move}\,(C,6,8),\ \mathtt{Del}\,(B,9),\ \mathtt{Del}\,(D,11),\\ & \mathtt{Ins}\,(C,12),\ \mathtt{Ins}\,(A,13),\ \mathtt{Ins}\,(D,16)\,\rangle\end{aligned}$$

and the edit distance between these sequences is

$$\begin{aligned}d_{\mathcal{S}}(\mathcal{S}_1,\mathcal{S}_2) \;=\; & \mathtt{c}\,(\mathtt{Move}\,(A,1,2)) + \mathtt{c}\,(\mathtt{Move}\,(B,3,5))\\ & +\mathtt{c}\,(\mathtt{Del}\,(A,4)) + \mathtt{c}\,(\mathtt{Move}\,(C,6,8))\\ & +\mathtt{c}\,(\mathtt{Del}\,(B,9)) + \mathtt{c}\,(\mathtt{Del}\,(D,11))\\ & +\mathtt{c}\,(\mathtt{Ins}\,(C,12)) + \mathtt{c}\,(\mathtt{Ins}\,(A,13))\\ & +\mathtt{c}\,(\mathtt{Ins}\,(D,16))\\ =\; & 0.5 \cdot 1 + 0.5 \cdot 2 + 1 + 0.5 \cdot 2 + 1\\ & +1 + 1 + 1 + 1 = 8.5.\end{aligned}$$

In this case, the cost of moving the event $(D, 11)$ to the time 16 is $\mathtt{c}\,(\mathtt{Move}\,(D,11,16)) = 2.5$. Because it is higher than the cost of first deleting and then inserting an event of type D, this Move-operation is not cost-effective, and thus, it is not a part of the optimal operation sequence.

If we want moving an event always to be preferred to deleting and inserting it, we can use the parameter value $V = \frac{1}{W} = \frac{1}{20} = 0.05$. Then the cost of moving an event is $\mathtt{c}\,(\mathtt{Move}\,(e_i, t_i, t_i^{'})) = 0.05 \cdot |\, t_i - t_i^{'}|$, and the cost of the maximum length move (20 time units) is $0.05 \cdot 20 = 1$. The optimal operation sequence $\hat{O}$ for the sequences $\mathcal{S}_1$ and $\mathcal{S}_2$ is now

$$\begin{aligned}\hat{O} \;=\; & \langle\, \mathtt{Move}\,(A,1,2),\ \mathtt{Move}\,(B,3,5),\ \mathtt{Del}\,(A,4),\\ & \mathtt{Move}\,(C,6,8),\ \mathtt{Del}\,(B,9),\ \mathtt{Ins}\,(C,12),\\ & \mathtt{Ins}\,(A,13),\ \mathtt{Move}\,(D,11,16)\,\rangle\end{aligned}$$

and the edit distance between the sequences is

$$\begin{aligned}d_{\mathcal{S}}(\mathcal{S}_1,\mathcal{S}_2) \;=\; & \mathtt{c}\,(\mathtt{Move}\,(A,1,2)) + \mathtt{c}\,(\mathtt{Move}\,(B,3,5))\\ & +\mathtt{c}\,(\mathtt{Del}\,(A,4)) + \mathtt{c}\,(\mathtt{Move}\,(C,6,8))\\ & +\mathtt{c}\,(\mathtt{Del}\,(B,9)) + \mathtt{c}\,(\mathtt{Ins}\,(C,12))\\ & +\mathtt{c}\,(\mathtt{Ins}\,(A,13)) + \mathtt{c}\,(\mathtt{Move}\,(D,11,16))\\ =\; & 0.05 \cdot 1 + 0.05 \cdot 2 + 1 + 0.05 \cdot 2 + 1\\ & +1 + 1 + 0.05 \cdot 5 = 4.50.\end{aligned}$$

Unlike in the situation above, it is now cost-effective to move the event $(D, 11)$ to the time 16.

Then let the numbers of occurrences of different event types in the set $\mathcal{E}$ and the alphabet-weighted costs of inserting and deleting events of these types, when $w(e) = occ(e)^{-1}$, be as in Table 4.1. If we now want moving of events for all event types to be cost-effective at least sometimes, the value of the parameter V should be less than $2 \cdot 0.01 = 0.02$. Assume first that we use the parameter value $V = 0.02$. Then the cost of moving an event (e_i, t_i) to time $t_i^{'}$ is $0.02 \cdot |\, t_i - t_i^{'}|$, and the cost of a move of the maximum length (20 time units) is 0.4. This means that moving an event is not always cost-effective. The edit distance between the sequences $\mathcal{S}_1$ and $\mathcal{S}_2$ with these parameter values is

$$\begin{aligned} d_{\mathcal{S}}(\mathcal{S}_1, \mathcal{S}_2) &= \mathsf{c}\,(\mathtt{Move}\,(A,1,2)) + \mathsf{c}\,(\mathtt{Move}\,(B,3,5)) \\ &\quad +\mathsf{c}\,(\mathtt{Del}\,(A,4)) + \mathsf{c}\,(\mathtt{Move}\,(C,6,8)) \\ &\quad +\mathsf{c}\,(\mathtt{Del}\,(B,9)) + \mathsf{c}\,(\mathtt{Del}\,(D,11)) \\ &\quad +\mathsf{c}\,(\mathtt{Ins}\,(C,12)) + \mathsf{c}\,(\mathtt{Ins}\,(A,13)) \\ &\quad +\mathsf{c}\,(\mathtt{Ins}\,(D,16)) \\ &= 0.02 \cdot 1 + 0.02 \cdot 2 + 0.01 + 0.02 \cdot 2 \\ &\quad +0.02 + 0.0125 + 0.05 + 0.01 + 0.0125 = 0.215. \end{aligned}$$

In this case moving the event $(D, 11)$ for 5 time units costs 0.10, but first deleting and then inserting an event of the type D costs only 0.025. Hence, deleting and inserting of the event is more cost-effective than moving it.

On the other hand, if we use a parameter value $V = (2 \cdot \mathsf{min\_w})/W$, the cost of moving an event is $\mathsf{c}\,(\mathtt{Move}\,(e_i, t_i, t_i^{'})) = [(2 \cdot 0.01)/20] \cdot |\, t_i - t_i^{'}| = 0.001 \cdot |\, t_i - t_i^{'}|$, and the cost of a Move-operation with the maximum length 20 is thus $0.001 \cdot 20 = 0.02$. The optimal operation sequence $\hat{O}$ is now different from the case with the parameter value $V = 2 \cdot \mathsf{min\_w}$, because moving the event $(D, 11)$ is cost-effective. The optimal operation sequence is now

$$\begin{aligned} \hat{O} &= \langle\, \mathtt{Move}\,(A,1,2),\ \mathtt{Move}\,(B,3,5),\ \mathtt{Del}\,(A,4), \\ &\quad \mathtt{Move}\,(C,6,8),\ \mathtt{Del}\,(B,9),\ \mathtt{Ins}\,(C,12), \\ &\quad \mathtt{Ins}\,(A,13),\ \mathtt{Move}\,(D,11,16)\,\rangle \end{aligned}$$

and the edit distance between the sequences $\mathcal{S}_1$ and $\mathcal{S}_2$ is

$$\begin{aligned} d_{\mathcal{S}}(\mathcal{S}_1, \mathcal{S}_2) &= \mathsf{c}\,(\mathtt{Move}\,(A,1,2)) + \mathsf{c}\,(\mathtt{Move}\,(B,3,5)), \\ &\quad \mathsf{c}\,(\mathtt{Del}\,(A,4)) + \mathsf{c}\,(\mathtt{Move}\,(C,6,8)) \\ &\quad \mathsf{c}\,(\mathtt{Del}\,(B,9)) + \mathsf{c}\,(\mathtt{Ins}\,(C,12)), \\ &\quad \mathsf{c}\,(\mathtt{Ins}\,(A,13)) + \mathsf{c}\,(\mathtt{Move}\,(D,11,16)) \\ &= 0.001 \cdot 1 + 0.001 \cdot 2 + 0.01 + 0.001 \cdot 2 \\ &\quad + 0.02 + 0.05 + 0.01 + 0.001 \cdot 5 = 0.100. \end{aligned}$$

□

Examples 4.11 and 4.12 clearly show that the costs of edit operations influence which operations are cost-effective, and thus, they determine what operation sequences are optimal, and what the edit distance between two sequences is. Therefore, it is important that the costs are chosen so that the edit distance between sequences corresponds to the intuitive notion of sequence similarity.

Assume now that two event sequences $\mathcal{S}_1$ and $\mathcal{S}_2$ have no events of similar type, i.e., there are no events $(e_i, t_i) \in \mathcal{S}_1$ and $(f_j, u_j) \in \mathcal{S}_2$ such that $e_i = f_j$. The edit distance between these sequences then is

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

Similarly, when two event type sequences S_1 and S_2 have no events of similar type, the edit distance between them is

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

The other extreme case when the edit distance between two sequences is zero is reached when the two sequences are identical. If one of the sequences is an empty sequence, then the distance between these two sequences is the sum of costs for inserting (deleting) all the events of the non-empty sequence. On the other hand, if both sequences compared are empty sequences, then their edit distance is zero by definition.

Example 4.13 Let $\mathcal{E} = \{A, B, C, D, E\}$ be the set of event types, and $\mathcal{O}_{\mathcal{S}}$ the set of edit operations. Then consider two event sequences $\mathcal{S}_3 = \langle (A, 6), (B, 8), (C, 12)\rangle$ and $\mathcal{S}_4 = \langle (D, 2), (E, 7), (D, 9)\rangle$ over $\mathcal{E}$. These two event sequences have no events of a similar type, and one optimal operation sequence $\hat{O}$ transforming the sequence $\mathcal{S}_3$ into the sequence $\mathcal{S}_4$ is

$$\begin{aligned} \hat{O} \;=\; &\langle\, \mathtt{Del}\,(A, 6),\; \mathtt{Del}\,(B, 8),\; \mathtt{Del}\,(C, 12), \\ &\;\mathtt{Ins}\,(D, 2),\; \mathtt{Ins}\,(E, 7),\; \mathtt{Ins}\,(D, 9)\,\rangle. \end{aligned}$$

The edit distance between these two event sequences with the unit costs is

$$\begin{aligned} d_{\mathcal{S}}(\mathcal{S}_3, \mathcal{S}_4) &= \mathtt{c}\,(\mathtt{Del}\,(A, 6)) + \mathtt{c}\,(\mathtt{Del}\,(B, 8)) + \mathtt{c}\,(\mathtt{Del}\,(C, 12)) \\ &\quad + \mathtt{c}\,(\mathtt{Ins}\,(D, 2)) + \mathtt{c}\,(\mathtt{Ins}\,(E, 7)) + \mathtt{c}\,(\mathtt{Ins}\,(D, 9)) \\ &= 1 + 1 + 1 + 1 + 1 + 1 = 6. \end{aligned}$$

On the other hand, if we use the alphabet-weighted operation costs given in Table 4.1, the optimal operation sequence is the same as above, and the edit distance between these two event sequences is

$$d_{\mathcal{S}}(\mathcal{S}_3, \mathcal{S}_4) = 0.01 + 0.02 + 0.05 + 0.0125 + 0.10 + 0.0125 = 0.205.$$

Similar results would be obtained, if we considered the two corresponding event type sequences $S_3 = \langle A, B, C \rangle$ and $S_4 = \langle D, E, D \rangle$, and used the set $\mathcal{O}_S$ of edit operations. $\square$

Now consider the set $\mathcal{E} = \{A, B, C, \ldots\}$ of event types, and two event sequences $\mathcal{S}_1 = \langle\, (e_1, t_1), \ldots, (e_m, t_m), (A, 57), (B, 58), (C, 60)\,\rangle$ and $\mathcal{S}_2 = \langle\, (A, 5), (B, 6), (C, 8), (f_1, u_1), \ldots, (f_m, u_m)\,\rangle$, where all the event types e_i and f_j are in the set $\mathcal{E} \setminus \{A, B, C\}$, and $e_i \neq f_j$ for all i, j. Then the edit distance $d_{\mathcal{S}}(\mathcal{S}_1, \mathcal{S}_2)$ is long because the sequences have so many mismatching events. Also the moves of any transformation would be very long: 52 time units. The sequence $\mathcal{S}_1$ is, however, more similar to the sequence

$$\begin{aligned}\mathcal{S}_3 \;=\; & \langle\, (g_1, v_1), (g_2, v_2), (g_3, v_3), (g_4, v_4), (A, 13), (B, 14), (C, 16), \\ & (g_5, v_5), \ldots, (g_m, v_m)\,\rangle\end{aligned}$$

than to the sequence $\mathcal{S}_2$, even though they all have a very similar subsequence. Namely, the types of the events in these subsequences are exactly the same and the relative differences between the occurrence times of the events of these subsequences in time are also the same in every case.

Then study the corresponding three event type sequences $S_1 = \langle\, A, B, C, e_1, \ldots, e_m\,\rangle$, $S_2 = \langle\, f_1, \ldots, f_m, A, B, C\,\rangle$, and $S_3 = \langle\, g_1, g_2, g_3, g_4, A, B, C, g_5, \ldots, g_m\,\rangle$. Now the occurrence times of the events do not influence the comparison of the sequences, and therefore, the sequence S_1 is as similar to the sequence S_2 as it is to the sequence S_3. This example shows that edit distances between event sequences can be very different from edit distances between the corresponding event type sequences. Therefore, if we want to emphasize similarity between entire sequences, not just parts of them, it is obvious that the occurrence times of events, not only their order, must be taken into account.

In all the three event sequences above, there is a subsequence where the time differences between events are the same. Also the three event type sequences have a short common subsequence. In some cases finding such common subsequences, either with or without occurrence times, might be useful; see [Gus97, SM97] for studies on this problem in molecular biology. Here we will not, however, discuss any further the problem of finding such local similarities between sequences.

4.2.4 Variations

The sets of edit operations can easily be extended to allow an edit operation that changes the type of an event, i.e., to allow a *substitution* of events, if such an operation is considered natural. Assume that we have a measure $h_{\mathcal{E}}$

of similarity defined on the set $\mathcal{E}$ of event types. Then a cost of transforming an event (e_i, t_i) into another event $(e_i^{'}, t_i^{'})$ in an event sequence, or changing an event of the type e in an event type sequence to an event of the type $e^{'}$, could be defined as $h_{\mathcal{E}}(e, e^{'}) + b$, where b is a constant.

The previous definition of the cost of substituting an event does not take into account the possible difference in the occurrence times t_i and $t_i^{'}$ in the event sequences. One solution to this problem might be to combine the costs of changing the event type and moving the event. In such a case, the total cost of substituting an event of one type by an event of another type should still be less than the sum of deletion and insertion costs of events, if we want substitutions of events to be cost-effective.

Assume then that we have two short sequences of events that have no events of similar type, and two long sequences that differ only with few events. Now the intuitive result would be that the edit distance between the long sequences is shorter than the edit distance between the short sequences. However, to transform one short sequence into another short sequence we need only a few operations. This, unfortunately, means that the edit distance between these sequences can be shorter than the edit distance between the two long sequences. If we want to avoid this phenomenon that intuitively seems incorrect, we have to take the lengths of the sequences into account when determining the edit distances. A simple way of doing this is to normalize the edit distances based on the costs of edit operations in the optimal operation sequence. In normalizing these edit distances we can use a factor

$$\sum_{(e_i,t_i)\in\mathcal{S}} \mathtt{c}\,(\mathtt{Del}\,(e_i, t_i)) + \sum_{(f_j,u_j)\in\mathcal{S}'} \mathtt{c}\,(\mathtt{Ins}\,(f_j, u_j))$$

when we consider the edit distance between two event sequences $\mathcal{S}$ and $\mathcal{S}^{'}$, and a factor

$$\sum_{e_i\in S} \mathtt{c}\,(\mathtt{Del}\,(e_i)) + \sum_{f_j\in S'} \mathtt{c}\,(\mathtt{Ins}\,(f_j))$$

when we determine the edit distance between two event type sequences S and $S^{'}$. Using this kind of normalization factors, the edit distances between sequences will vary between zero and one.

4.3 Algorithms for computing event sequence similarity

In this section we present algorithms for computing the edit distance between two sequences and for computing all the pairwise edit distances for a set $\mathcal{SI}$ of interesting sequences. The algorithms are mainly presented by using event sequences, but they all also apply to the case of event type sequences. We start by describing an algorithm for computing the weighted edit distance between two event sequences.

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

Given two event 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 weighted edit distance between the event 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 $\mathcal{S}_2$. A recursive relationship between the value of $\mathsf{r}(i,j)$ for all positive indexes i and j and the values of r with smaller indexes than i and j is given by the recurrence relation [Gus97]. When there are no smaller indexes, the value of $\mathsf{r}(i,j)$ must defined explicitly by the *base conditions* for $\mathsf{r}(i,j)$. In this thesis, we use the following base conditions and recurrence relation.

Definition 4.10 Let $\mathcal{S}_1 = \langle (e_1,t_1),(e_2,t_2),\ldots,(e_m,t_m)\rangle$ and $\mathcal{S}_2 = \langle (f_1,u_1),(f_2,u_2),\ldots,(f_n,u_n)\rangle$ be two event sequences, or similarly $S_1 = \langle e_1,e_2,\ldots,e_m\rangle$ and $S_2 = \langle f_1,f_2,\ldots,f_n\rangle$ two event type sequences. The base conditions and the recurrence relation for the value $\mathsf{r}(i,j)$ are

$$\begin{array}{rcl}
\mathsf{r}(0,0) & = & 0 \\
\mathsf{r}(i,0) & = & \mathsf{r}(i-1,0) + w(e),\ i>0 \\
\mathsf{r}(0,j) & = & \mathsf{r}(0,j-1) + w(f),\ j>0 \\
\mathsf{r}(i,j) & = & min\,\{\,\mathsf{r}(i-1,j)+w(e),\ \mathsf{r}(i,j-1)+w(f), \\
& & \mathsf{r}(i-1,j-1)+\mathsf{k}(i,j)\,\},\ i>0,\ j>0
\end{array}$$

where $w(e)$ and $w(f)$ are costs of deleting (inserting) an event e_i or f_j, respectively. For event sequences we define $\mathsf{k}(i,j)$ as

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

In the case of event type sequences, $\mathsf{k}(i,j)$ is defined as

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

□

The second component of the dynamic programming approach is using the base conditions and the recurrence relation to compute the value $\mathsf{r}(m,n)$, i.e., the edit distance between the event sequences $\mathcal{S}_1$ and $\mathcal{S}_2$. We use a *bottom-up* approach [Gus97] to compute such a distance. This means the values $\mathsf{r}(i,j)$ are computed for increasing values of indexes and saved in an $(m+1) \times (n+1)$ dynamic programming table.

Often the dynamic programming table is filled in one column at a time, in order of increasing index i. In our approach, however, we fill the table in one row at a time, in order of increasing index j; the resulting table is the same in both cases. 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 already know the table values $\mathsf{r}(i-1,j)$, $\mathsf{r}(i,j-1)$ and $\mathsf{r}(i-1,j-1)$. The value of $\mathsf{r}(i,j)$ is obtained by adding to $\mathsf{r}(i-1,j)$ the cost of deletion of an event (e_i,t_i) from the sequence $\mathcal{S}_1$, by adding to $\mathsf{r}(i,j-1)$ the cost of insertion of an event (f_j,u_j) to the sequence $\mathcal{S}_2$, or by adding to $\mathsf{r}(i-1,j-1)$ the cost $\mathsf{k}(i,j)$ of transforming an event (e_i,t_i) in the sequence $\mathcal{S}_1$ into an event (f_j,u_j) in the sequence $\mathcal{S}_2$. The cost $\mathsf{k}(i,j)$ depends on whether $e_i = f_j$ or not. Because $\mathsf{r}(i,j)$ has to be the minimum cost of transforming the first i events of the sequence $\mathcal{S}_1$ into 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 the set $\mathcal{E} = \{A,B,C,D,E\}$ of event types, two event sequences

$$\mathcal{S}_1 = \langle (A,1),(B,3),(A,4),(C,6),(B,9),(D,11) \rangle$$

and

$$\mathcal{S}_2 = \langle (A,2),(B,5),(C,8),(C,12),(A,13),(D,16) \rangle,$$

and the operation set $\mathcal{O}_{\mathcal{S}}$. Assume further that we have the unit costs for the `Ins`- and `Del`-operations, i.e., $\mathsf{c}\,(\texttt{Ins}\,(e_i,t_i)) = \mathsf{c}\,(\texttt{Del}\,(e_i,t_i)) = 1$, for every event type in the set $\mathcal{E}$, and in computing the costs of the `Move`-operations we use the parameter value $V = 0.5$.

The dynamic programming table r used in the computation of the edit distance between the event sequences $\mathcal{S}_1$ and $\mathcal{S}_2$ is given in Figure 4.4. The value $\mathsf{r}(3,5)$ in the table, for instance, is 5.5. The third event of the sequence $\mathcal{S}_1$ and the fifth event of the sequence $\mathcal{S}_2$ are both of the type A. The value of $\mathsf{r}(3,5)$ can be obtained either from $\mathsf{r}(3,4)$ by inserting an

r	j	0	1	2	3	4	5	6
i			(A,2)	(B,5)	(C,8)	(C,12)	(A,13)	(D,16)
0		0	1	2	3	4	5	6
1	(A,1)	1	0.5	1.5	2.5	3.5	4.5	5.5
2	(B,3)	2	1.5	1.5	2.5	3.5	4.5	5.5
3	(A,4)	3	2.5	2.5	3.5	4.5	5.5	6.5
4	(C,6)	4	3.5	3.5	3.5	4.5	5.5	6.5
5	(B,9)	5	4.5	4.5	4.5	5.5	6.5	7.5
6	(D,11)	6	5.5	5.5	5.5	6.5	7.5	8.5

Figure 4.4: The dynamic programming table used to compute the edit distance between event sequences $\mathcal{S}_1$ and $\mathcal{S}_2$ in Example 4.14.

event of type A, or from $\mathsf{r}(2,5)$ by deleting an event of type A. The value cannot, however, be obtained from $\mathsf{r}(2,4)$ because adding $\mathsf{k}(3,5) = 4.5$ to $\mathsf{r}(2,4)$ would be more than the value with the other two alternatives. □

The weighted edit distance between two event 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 4.10. Its input are the two event sequences and the parameter values for computing the costs of edit operations: the value of the parameter V and the values $w(e)$ for each $e \in \mathcal{E}$. The output of the algorithm is the edit distance between the two sequences. If this algorithm is applied to computing edit distances between event type sequences, the input of the algorithm are the two event type sequences and the values $w(e)$ for each $e \in \mathcal{E}$.

Algorithm 4.1 Edit distance between event sequences

Input: Two event sequences $\mathcal{S}_1$ and $\mathcal{S}_2$, and values $w(e)$ for each $e \in \mathcal{E}$ and a value of the parameter V.

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

Method:

$\mathsf{r}(0,0) = 0$;

for $i = 1$ **to** m **do**

  $\mathsf{r}(i,0) = \mathsf{r}(i-1,0) + w(e)$; **od**;

for $j = 1$ **to** n **do**

  $\mathsf{r}(0,j) = \mathsf{r}(0,j-1) + w(f)$; **od**;

```
for i = 1 to m do
    for j = 1 to n do
        r(i,j) = min { r(i-1,j) + w(e),
                       r(i,j-1) + w(f),
                       r(i-1,j-1) + k(i,j) };
    od;
od;
output r(m,n);
```

To extract the optimal operation sequence leading to the computed edit distance, we can add a traceback method to Algorithm 4.1. 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 $\mathsf{r}(i,j)$ is computed, we set a pointer from the cell (i,j) to the cell $(i,j-1)$ if $\mathsf{r}(i,j) = \mathsf{r}(i,j-1) + w(f)$, a pointer from the cell (i,j) to the cell $(i-1,j)$ if $\mathsf{r}(i,j) = \mathsf{r}(i-1,j) + w(e)$, 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 the zeroth column a pointer to the cell just above it. In all the other cells, it is possible (and common) to have 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, u_j) into $\mathcal{S}_2$, and each vertical pointer as a deletion of an event (e_i, t_i) from $\mathcal{S}_1$. Each diagonal edge is interpreted as a move,

r	j	0	1	2	3	4	5	6
i								
0		0	$\leftarrow$ 1	$\leftarrow$ 2	$\leftarrow$ 3	$\leftarrow$ 4	$\leftarrow$ 5	$\leftarrow$ 6
1		$\uparrow$ 1	$\nwarrow$ 0.5	$\leftarrow$ 1.5	$\leftarrow$ 2.5	$\leftarrow$ 3.5	$\leftarrow$ 4.5	$\leftarrow$ 5.5
2		$\uparrow$ 2	$\uparrow$ 1.5	$\nwarrow$ 1.5	$\leftarrow$ 2.5	$\leftarrow$ 3.5	$\leftarrow$ 4.5	$\leftarrow$ 5.5
3		$\uparrow$ 3	$\uparrow$ 2.5	$\uparrow$ 2.5	$\nwarrow\leftarrow\uparrow$ 3.5	$\nwarrow\leftarrow\uparrow$ 4.5	$\leftarrow\uparrow$ 5.5	$\nwarrow\leftarrow\uparrow$ 6.5
4		$\uparrow$ 4	$\uparrow$ 3.5	$\uparrow$ 3.5	$\nwarrow$ 3.5	$\leftarrow$ 4.5	$\leftarrow$ 5.5	$\leftarrow$ 6.5
5		$\uparrow$ 5	$\uparrow$ 4.5	$\uparrow$ 4.5	$\uparrow$ 4.5	$\nwarrow\leftarrow\uparrow$ 5.5	$\nwarrow\leftarrow\uparrow$ 6.5	$\nwarrow\leftarrow\uparrow$ 7.5
6		$\uparrow$ 6	$\uparrow$ 5.5	$\uparrow$ 5.5	$\uparrow$ 5.5	$\nwarrow\leftarrow\uparrow$ 6.5	$\nwarrow\leftarrow\uparrow$ 7.5	$\leftarrow\uparrow$ 8.5

Figure 4.5: A dynamic programming table used to compute the edit distance between event sequences $\mathcal{S}_1$ and $\mathcal{S}_2$ in Example 4.14 with pointers for extracting the optimal operation sequence $\hat{O}$.

if the corresponding event types match ($e_i = f_j$), and as a deletion and an insertion if the event types 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 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 corresponding to different optimal operation sequences.

The traceback of an optimal operation sequence can be done with Algorithm 4.2. This algorithm is given the dynamic programming table with pointers as input, and it outputs one optimal operation sequence. A call to this algorithm can be added to Algorithm 4.1 before Line 11, i.e., before outputting the edit distance.

Algorithm 4.2 Extracting an optimal operation sequence

Input: A dynamic programming table r used to compute the distance between event sequences $\mathcal{S}_1$ and $\mathcal{S}_2$.

Output: An optimal operation sequence transforming the event sequence $\mathcal{S}_1$ into the event sequence $\mathcal{S}_2$.

Method:

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

For extracting an optimal operation sequence of transforming an event type sequence into another, Line 5 of Algorithm 4.2 should simply be omitted.

Assume then that we have a set $\mathcal{SI}$ of interesting sequences, instead of just two sequences, and we want to find out similarities between the sequences in this set. This means that we have to compute the edit distances between these sequences. If the sequences in the set $\mathcal{SI}$ are event sequences, these distances can be computed using Algorithm 4.3. The input of this algorithm is the set $\mathcal{SI}$ of interesting event sequences, and its output are the pairwise edit distances between the sequences in this set. The kind of edit operation costs that are used in determining the edit distances is omitted from this algorithm, but when implementing the algorithm they should be taken into account. Given a set $\mathcal{SI}$ of event type sequences, a similar algorithm could also be used to compute pairwise edit distances between these event type sequences.

Algorithm 4.3 Distances between a set of event sequences
Input: A set $\mathcal{SI}$ of interesting event sequences.
Output: Pairwise edit distances between the sequences of the set $\mathcal{SI}$.
Method:

for all sequence pairs $(\mathcal{S}_i, \mathcal{S}_j)$ where $\mathcal{S}_i$ and $\mathcal{S}_j \in \mathcal{SI}$ **do**
  calculate $d_{\mathcal{S}}(\mathcal{S}_i, \mathcal{S}_j)$;
od;
output the pairwise edit distances $d_{\mathcal{S}}(\mathcal{S}_i, \mathcal{S}_j)$;

Complexity considerations

In Algorithm 4.1 the size of the dynamic programming table is $(m + 1) \times (n + 1)$, where m and n are the lengths of the sequences $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively. Filling in one cell of the table requires a constant number of cell examinations, arithmetic operations and comparisons. Therefore, the time and the space complexities of the algorithm are both $O(mn)$. If the sequences are fairly short, the quadratic behavior of Algorithm 4.1 is not a problem. However, if the sequences are typically very long, we can use more efficient algorithms than just dynamic programming to compute similarities between event type sequences; see [Ste94] for some examples of such algorithms. Unfortunately, there are not yet the same kind of efficient algorithms for computing similarities between event sequences.

On each iteration in Algorithm 4.2 either the index i, the index j, or both of them are decremented. This means that the maximum number of iterations is $m + n$, and that the time complexity of the extraction of the optimal operation sequence is $O(m+n)$. Because this algorithm uses a similar dynamic programming table to Algorithm 4.1, the space complexity of the algorithm is $O(mn)$.

Assume that we have a set $\mathcal{SI}$ of sequences. When there are $|\mathcal{SI}|$ sequences in this set, the number of the pairwise edit distances computed by Algorithm 4.3 is $\binom{|\mathcal{SI}|}{2}$. Each edit distance computation with Algorithm 4.1 takes $O(mn)$ time, and therefore, the time complexity of Algorithm 4.3 is $O(|\mathcal{SI}|^2 mn)$. When each edit distance between two sequences is computed with Algorithm 4.1, and the pairwise edit distances are first output when they all have been computed, the space complexity of Algorithm 4.3 is $O(mn + |\mathcal{SI}|^2)$.

4.4 Experiments

In this section we present some results of experiments on similarity between event sequences. In Section 4.4.1 we describe the data sets used in these experiments. After that, the results obtained with these data sets are presented in Section 4.4.2. All these experiments were run on a PC with a 233 MHz Pentium processor and a 64 MB main memory, under the Linux operating system.

4.4.1 Data sets

The experiments on similarity between event sequences use two real-life data sets: an alarm data set from a Finnish telecommunication company, and a log of WWW page requests collected at the Department of Computer Science at the University of Helsinki. Both these data sets resided in flat text files.

Telecommunication alarm data

The telecommunication data set consists of 73 679 alarms. The data was collected during 50 days, i.e., a time period covering over seven weeks. On three days there were no alarms at all, and on one day a total of 10 277 alarms. The number of different alarm types in this data set is 287. The numbers of occurrences of alarm types vary a lot: from one occurrence to 12 186 occurrences, with an average of 257 occurrences per alarm type.

We selected several alarm types from the set of 287 alarm types. For each of those alarm types we extracted all the subsequences preceding their

Interesting alarm type	Number of occurrences	Number of non-empty preceding subsequences		
		$W = 30$	$W = 60$	$W = 120$
1400	9	6	9	9
2402	16	14	14	14
7272	10	10	10	10
7712	101	83	88	92

Table 4.2: The numbers of occurrences of the chosen four alarm types and their non-empty preceding subsequences with the chosen window widths.

occurrences using three window widths. That is, we extracted subsequences where the alarms occurred within 30, 60 or 120 seconds at the most before the occurrences of the given alarm type. For each selected alarm type we experimented with two sets of preceding subsequences: a set of event type sequences and a set of event sequences.

In the following we present results obtained with four alarm types that represent different groups of alarm types, namely, the alarm types 1400, 2402, 7272 and 7712. Table 4.2 presents, for each of these alarm types, the numbers of occurrences of these four alarm types and the numbers of the non-empty subsequences preceding them with the chosen three window widths W. The set of event subsequences preceding alarms of the alarm type 1400 with the window width W of 60 seconds is shown in Figure 4.6. In this set there are nine sequences whose lengths are at least two alarms, i.e., all these sequences are non-empty sequences. Note that the number of empty preceding subsequences depends on both the window width and the interesting alarm type.

WWW log data

The log of WWW page requests used in our experiments was collected during twelve days at the University of Helsinki from the requests for the pages in the WWW server of the Department of Computer Science. We first filtered out from the raw data all the unsuccessful page requests and all the references to pictures located at pages, because we wanted to concentrate on the successful page requests. After that, the data set still consists of 45 322 successful page requests for 6 023 different WWW pages. The number of requests per page varies from one to 1 848, with a mean of 7.5 requests. A total of 2 563 pages are requested only once. The number of requesting hosts is 7 633, and the number of requests from one

$\mathcal{S}_1$	$=$	$\langle(690,43),(1001,43),(690,47),(1567,49),(1567,49),(1567,49),$ $(1567,49),(690,51)\rangle$
$\mathcal{S}_2$	$=$	$\langle(690,38),(1001,38)\rangle$
$\mathcal{S}_3$	$=$	$\langle(1553,8),(691,39),(690,39),(690,39),(1001,39)\rangle$
$\mathcal{S}_4$	$=$	$\langle(2263,19),(7161,28),(7161,28),(7161,28),(691,38),(690,38),$ $(690,38),(1241,38),(1001,38),(2263,44)\rangle$
$\mathcal{S}_5$	$=$	$\langle(691,39),(690,39),(690,39),(1001,39)\rangle$
$\mathcal{S}_6$	$=$	$\langle(7161,27),(7161,27),(691,38),(690,38),(690,38),(1001,38),$ $(421,58)\rangle$
$\mathcal{S}_7$	$=$	$\langle(1241,15),(1903,24),(1241,25),(7161,28),(7161,28),(1578,32),$ $(691,39),(690,39),(690,39),(1001,39),(1585,54)\rangle$
$\mathcal{S}_8$	$=$	$\langle(7161,28),(7161,28),(2250,35),(691,38),(1001,38),(690,39),$ $(690,39)\rangle$
$\mathcal{S}_9$	$=$	$\langle(2470,28),(2250,28),(691,29),(690,29),(690,29),(1001,29),$ $(7705,38)\rangle$

Figure 4.6: The set of subsequences preceding alarms of the type 1400 with the window width $W = 60$. The occurrence times of alarms in the sequences present how many seconds before the alarm of the type 1400 they occurred.

host varies from one to 4 656 requests. A total of 4 189 hosts have made only one request, whereas the mean of requests per host is six requests.

We selected from this data set several WWW pages. For each page we extracted all subsequences that preceded requests for the given page with the same requesting host. In extracting these subsequences we used window widths of one, five and ten minutes, i.e., 60, 300 or 600 seconds. For each selected page we experimented with two sets of preceding subsequences: a set of event type sequences and a set of event sequences.

In this thesis we present results for sets of sequences preceding requests for four chosen WWW pages. The *CSUH project* page is the page describing different research projects in the yearly report of the Department of Computer Science, the *Data mining* page the main page of the Data Mining research group, the *KDD link* page the page of KDD related links on the Data Mining research group page, and the *"What is Linux"* page the page that briefly describes the Linux operating system and its birth. Table 4.3 shows the numbers of requests for these pages as well as the numbers of the non-empty subsequences preceding these requests with the chosen window widths. Figure 4.7 presents the event subsequences preceding requests

Interesting WWW page	Number of occurrences	Number of non-empty preceding subsequences		
		$W = 60$	$W = 300$	$W = 600$
CSUH project	24	8	10	11
Data mining	88	37	41	42
KDD link	15	12	15	15
"What is Linux"	96	37	61	65

Table 4.3: The numbers of requests for the chosen four WWW pages and their non-empty preceding subsequences with the chosen window widths.

$$
\begin{aligned}
\mathcal{S}_1 &= \langle(\text{../abstracts/km-pskd-94.ps.Z}, 33)\rangle \\
\mathcal{S}_2 &= \langle(\text{../dm/group.html}, 23), (\text{../aaps/Eliot/uutta.html}, 44)\rangle \\
\mathcal{S}_3 &= \langle(\text{../dm/publications.html}, 18)\rangle \\
\mathcal{S}_4 &= \langle(\text{../dm/toc.html}, 19), (\text{../dm/logo.html}, 19), \\
&\quad (\text{../dm/datamine.html}, 19), (\text{../dm/counter.html}, 19), \\
&\quad (\text{../dm/}, 19)\rangle \\
\mathcal{S}_5 &= \langle\rangle \\
\mathcal{S}_6 &= \langle(\text{../dm/counter.html}, 26), (\text{../dm/logo.html}, 27), \\
&\quad (\text{../dm/toc.html}, 29), (\text{../dm/datamine.html}, 29)\rangle \\
\mathcal{S}_7 &= \langle(\text{../dm/datamine.html}, 36), (\text{../dm/toc.html,58}), \\
&\quad (\text{../dm/logo.html}, 58), (\text{../dm/counter.html,58})\rangle \\
\mathcal{S}_8 &= \langle(\text{../dm/publications.html}, 29), (\text{/cgi-bin/form-mailer.pl/}, 39)\rangle \\
\mathcal{S}_9 &= \langle(\text{../dm/comments.html}, 15)\rangle \\
\mathcal{S}_{10} &= \langle(\text{../dm/publications.html}, 53)\rangle \\
\mathcal{S}_{11} &= \langle(\text{../mannila/postscripts/sigmodworkshop.ps}, 47)\rangle \\
\mathcal{S}_{12} &= \langle(\text{../dm/toc.html}, 23), (\text{../dm/logo.html}, 23), \\
&\quad (\text{../dm/datamine.html}, 23), (\text{../dm/counter.html}, 23), \\
&\quad (\text{../dm/}, 24), (\text{/research/}, 60)\rangle \\
\mathcal{S}_{13} &= \langle\rangle \\
\mathcal{S}_{14} &= \langle\rangle \\
\mathcal{S}_{15} &= \langle(\text{../dm/comments.html}, 5), (\text{../mannila/dm-publications.html}, 28), \\
&\quad (\text{../mannila/luokiteltu.ps}, 35)\rangle
\end{aligned}
$$

Figure 4.7: The set of event subsequences preceding requests for the KDD link page with the window width $W = 60$. The occurrence times of requests in the sequences present how many seconds before the request for the KDD link page they occurred.

for the KDD link page using the window width W of 60 seconds[3]. This set contains fifteen sequences, of which three are empty sequences. Note that in this case the number of empty preceding subsequences also depends largely on the window width and the interesting WWW page.

4.4.2 Results

In our experiments we computed edit distances between sequences in each test set both with the unit and the alphabet-weighted costs. These operation costs were defined according to Definition 4.10 so that with the unit costs $w(e) = 1$ and with the alphabet-weighted costs $w(e) = occ(e)^{-1}$. The exact numbers of the occurrences of each alarm type were obtained from the whole telecommunication alarm sequence, and the number of requests for each WWW page from the WWW log of successful page requests.

The basic value for the window width W in the alarm data set was 60 seconds and in the WWW data set 600 seconds. In addition, some experiments on the sets of alarm sequences were done with window widths $W = 30$ and $W = 120$, on the sets of WWW page request sequences with window widths $W = 60$ and $W = 300$.

For the parameter V we used the basic value $\frac{1}{W}$ with the unit operation costs and $\frac{2\cdot\mathsf{min\_w}}{W}$ with the alphabet-weighted operation costs. With these parameter values, moving an event is always preferred to deleting and inserting one. We also wanted to study the influence of the parameter V on the edit distances. Therefore, we made additional experiments with other values of the parameter V. The parameter values used with alarm sequences are given in Table 4.4, and the values used with WWW page request sequences in Table 4.5.

In all our experiments, the lengths of sequences varied a lot. As discussed in Section 4.2, computing the basic edit distance between sequences may in such a case lead to a situation where two sequences that have no common events at all have a short distance, whereas two sequences that contain quite a long common subsequence have a long distance. To eliminate the influence of the lengths of the sequences on the distances, we normalized the edit distances. This normalization means that the basic value of the edit distance between the event sequences $\mathcal{S}$ and $\mathcal{S}'$ was divided by the sum of the operation costs of deleting all the events of the event sequence $\mathcal{S}$ and inserting all the events of the event sequence $\mathcal{S}'$, i.e., by the factor $\sum_{(e_i,t_i)\in\mathcal{S}} \mathtt{c}\,(\mathtt{Del}\,(e_i,t_i)) + \sum_{(f_j,u_j)\in\mathcal{S}'} \mathtt{c}\,(\mathtt{Ins}\,(f_j,u_j))$. Similarly, the basic value of the edit distance between the event type sequences S and

[3]We chose to present the subsequences with the shortest chosen window width, since some of the subsequences with the other two window widths were extremely long.

Type of operation costs	Values of the parameter V
Unit costs	$\frac{1}{W}$, $\frac{2}{W}$, $\frac{4}{W}$, $\frac{8}{W}$, $\frac{24}{W}$, $\frac{120}{W}$
Alphabet-weighted costs	$\frac{2\cdot\mathsf{min\_w}}{W}$, $\frac{4\cdot\mathsf{min\_w}}{W}$, $\frac{8\cdot\mathsf{min\_w}}{W}$, $\frac{16\cdot\mathsf{min\_w}}{W}$, $\frac{48\cdot\mathsf{min\_w}}{W}$, $2\cdot\mathsf{min\_w}$

Table 4.4: Values of the parameter V used in alarm sequence experiments.

Type of operation costs	Values of the parameter V
Unit costs	$\frac{1}{W}$, $\frac{2}{W}$, $\frac{4}{W}$, $\frac{8}{W}$, $\frac{24}{W}$, $\frac{60}{W}$, $\frac{120}{W}$
Alphabet-weighted costs	$\frac{2\cdot\mathsf{min\_w}}{W}$, $\frac{4\cdot\mathsf{min\_w}}{W}$, $\frac{8\cdot\mathsf{min\_w}}{W}$, $\frac{16\cdot\mathsf{min\_w}}{W}$, $\frac{48\cdot\mathsf{min\_w}}{W}$, $\frac{120\cdot\mathsf{min\_w}}{W}$, $\frac{240\cdot\mathsf{min\_w}}{W}$, $2\cdot\mathsf{min\_w}$

Table 4.5: Values of the parameter V used in WWW log experiments.

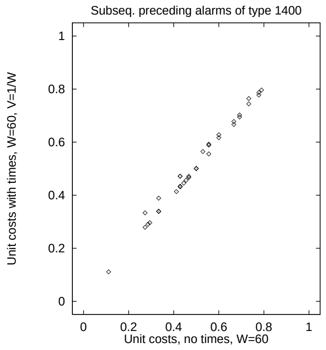
S' was divided by the sum of the operation costs of deleting all the events of the event type sequence S and inserting all the events of the event type sequence S', i.e., by the factor $\sum_{e_i \in S} \mathsf{c}\,(\mathtt{Del}\,(e_i)) + \sum_{f_j \in S'} \mathsf{c}\,(\mathtt{Ins}\,(f_j))$. In this way all the edit distances between sequences were scaled between zero and one.

All the edit distances were computed using the algorithms of Section 4.3. Similarly to Section 3.5.2, we here give only a few examples of the actual edit distances. Because the actual distance values are often irrelevant, we mainly concentrate on the orders of the edit distances given by the different measures.

Event sequences versus event type sequences

We started our experiments by studying how edit distances between event type sequences are related to edit distances between event sequences. In these experiments we used the basic values of the parameters W and V (see page 91).

Figure 4.8 illustrates how edit distances between event type sequences and event sequences preceding alarms of the type 1400 are related to each other. The plot in Figure 4.8a describes the relationship between edit distances with the unit costs, and the plot in Figure 4.8b the relationship between edit distances with the alphabet-weighted costs. In both cases

a) 

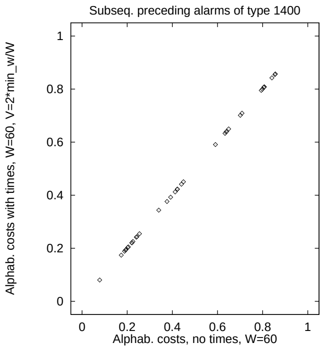
b) 

Figure 4.8: Comparison of edit distances between event type sequences and event sequences preceding alarms of the type 1400 within 60 seconds when a) the unit operations costs with $V = \frac{1}{W}$, and b) the alphabet-weighted costs with $V = \frac{2 \cdot \mathsf{min\_w}}{W}$ are used.

the edit distances are positively linearly correlated, and the orders of the distances are nearly the same. Also when a pair $(\mathcal{S}, \mathcal{S}^{'})$ of event sequences has the shortest edit distance $d_{\mathcal{S}}$, the corresponding event type sequence pair $(S, S^{'})$ has the shortest edit distance d_S. This is true for both the unit and the alphabet-weighted operation costs. The measures d_S and $d_{\mathcal{S}}$ also agree on the pair of sequences that have the longest edit distance with both types of edit operation costs. The same kind of phenomena could also be observed with the sets of alarm sequences preceding occurrences of the other chosen alarm types with the window width $W = 60$.

Corresponding experiments were also made with the sets of sequences from the WWW log. Figure 4.9 presents how the edit distances between event type sequences preceding the requests for the KDD link page are related to the edit distances between the corresponding event sequences. In this case the measures d_S and $d_{\mathcal{S}}$ are also positively correlated. Besides, these measures agree both on the pairs of sequences with the shortest and the longest edit distances, regardless of the type of operation costs used. Actually, with both types of operation costs, the longest edit distance between pairs of sequences is one, and there are a total of 24 such pairs of sequences, both event sequences and event type sequences. The same holds good for the experiments with the other test sets of WWW page request sequences. In the case of the sequences preceding requests for the "What is Linux" page the edit distances are also positively correlated. However, as Figure 4.10 shows, in this case there are many pairs of event type sequences that have the same edit distance, whereas the edit distances between the corresponding event sequences vary a lot.

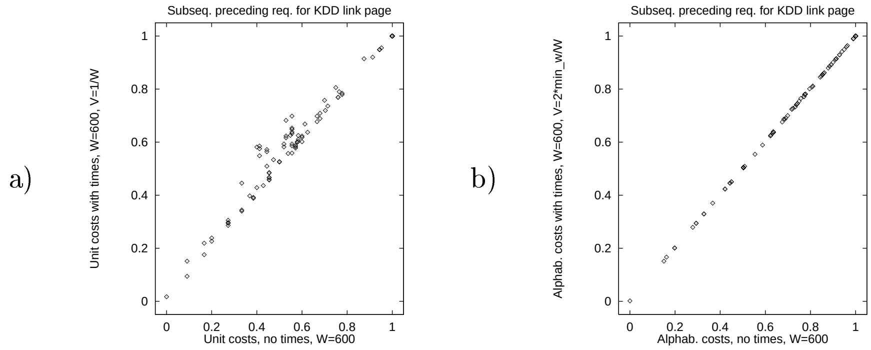


Figure 4.9: Comparison of edit distances between event type sequences and event sequences preceding requests for the KDD link page within 600 seconds when a) the unit operation costs with $V = \frac{1}{W}$, and b) the alphabet-weighted operation costs with $V = \frac{2 \cdot \mathsf{min\_w}}{W}$ are used.

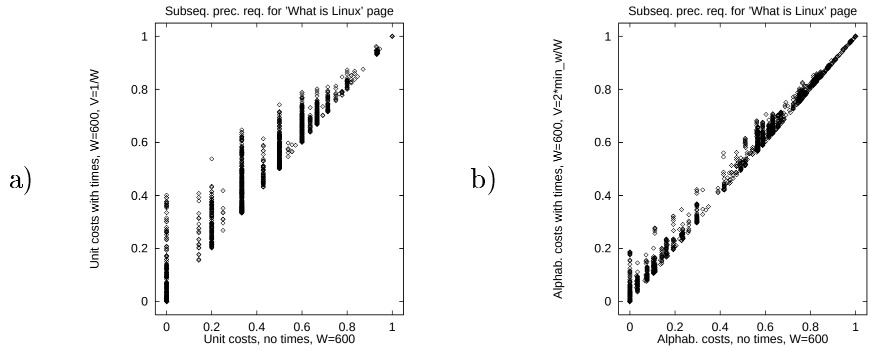


Figure 4.10: Comparison of edit distances between event type sequences and event sequences preceding requests for the "What is Linux" page within 600 seconds when a) the unit operation costs with $V = \frac{1}{W}$, and b) the alphabet-weighted operation costs with $V = \frac{2 \cdot \mathsf{min\_w}}{W}$ are used.

The strong correlation between the edit distances between event type sequences and event sequences with the basic values of the parameters W and V might lead us to a conclusion that with these parameter values it is enough to restrict our attention only to event type sequences, or event sequences. However, when we looked at the actual edit distances more closely, we found that the situation was not that simple. There were namely several pairs of event type sequences that had exactly the same edit distance, but when we took into account the occurrence times of the events, the edit distances between the corresponding event sequences differed from each other.

In the set of event type sequences preceding alarms of the type 1400, for example, there are five pairs of sequences whose edit distance with the unit costs is 0.4286, whereas the edit distances between the corresponding pairs of event sequences with the unit costs vary between 0.4317 and 0.4714. Similar examples could be found from any of the test sets considered. The differences between the edit distances of pairs of event sequences depend on how long moves of events we need to make. If the moves are short, the differences between the distances are also small, whereas the differences can be remarkable, if the lengths of the moves vary a lot. Because the edit distances between event sequences more clearly find differences between the pairs of sequences than the edit distances between event type sequences, it should be clear that taking the occurrence times of the events into account is often important. Still, edit distances between event type sequences should not be disregarded completely.

Unit versus alphabet-weighted operation costs

We also compared how edit distances between sequences change when different operation costs are used. In these experiments we again used the basic values of the parameters W and V (see page 91).

Figure 4.11 describes how the edit distances $d_{\mathcal{S}}$ of event subsequences preceding alarms of the type 1400, 2402, 7272 and 7712 are related to each other when different types of operation costs are used; the results for the sets of corresponding event type sequences are very similar. The distributions of points in these four plots are wide, indicating that the operation costs really have an influence on how similar two event sequences are. In the case of the sequences preceding alarms of the types 1400 and 7272, the measures with different types of operation costs do not agree on the most similar or the most dissimilar pair of sequences. With alarm types 2402 and 7712, the situation is somewhat different. Among the sequences preceding alarms of the type 2402, one pair of empty sequences is completely similar with both the unit and the alphabet-weighted operation costs, but with both types of operations costs the pair of sequences that has the second shortest distance is also the same. On the other hand, in this set of sequences 71 pairs of sequences are completely dissimilar, i.e., their edit distance is one, with both types of operation costs. In the set of sequences preceding alarms of the type 7712, 80 pairs of sequences are completely similar, and a total of 2 867 pairs of sequences are completely dissimilar, with both types of the operation costs. So, in these two cases, the edit distances with different types of operation costs agree on the most

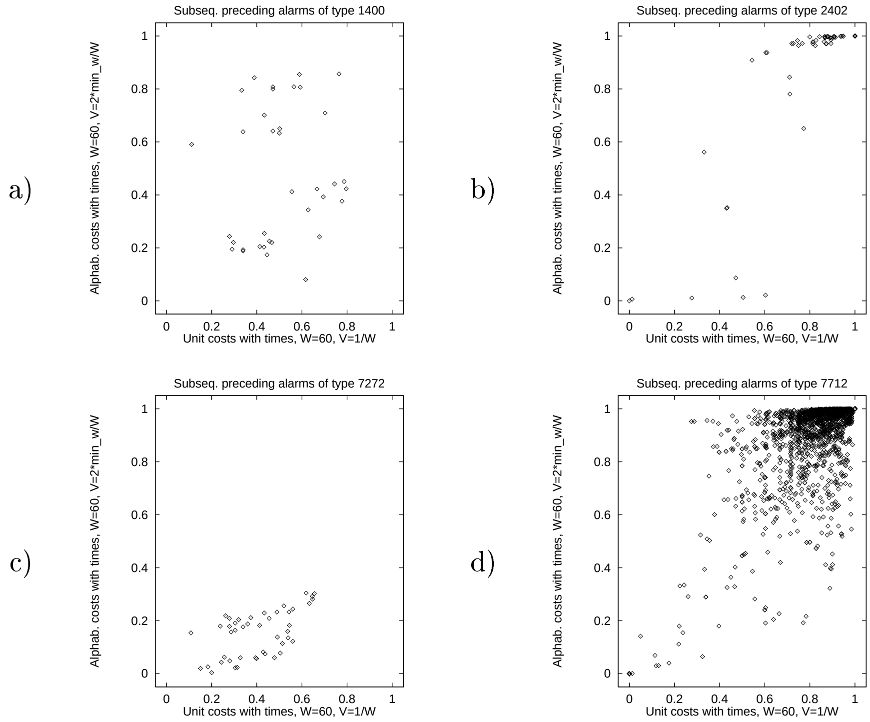


Figure 4.11: Comparison of edit distances between event sequences preceding alarms of the type a) 1400, b) 2402, c) 7272, and d) 7712 with the window width $W = 60$ using the unit costs with $V = \frac{1}{W}$ and the alphabet-weighted costs with $V = \frac{2 \cdot \mathsf{min\_w}}{W}$.

similar pair and the most dissimilar pair of sequences. Despite this, the distributions of the edit distances in Figures 4.11b and 4.11d show that it makes a difference what type of operation costs are used in computing the edit distances.

This same conclusion can also be made by studying the orders of the edit distances obtained with the different types of operation costs. Namely, the actual edit distances show that two sequences, that have a rather short edit distance with the unit costs, can have a long edit distance with the alphabet-weighted costs. This phenomenon also holds good in the other direction: even if the edit distance between two sequences is short with the alphabet-weighted costs, their edit distance with the unit costs can be much longer. Consider, for example, the set of event sequences preceding an alarm of the type 1400 given in Figure 4.6, and especially the event sequences $\mathcal{S}_2$, $\mathcal{S}_3$ and $\mathcal{S}_5$. The edit distance between the event sequences $\mathcal{S}_2$

and $\mathcal{S}_5$ is 0.3389 with the unit costs, but with the alphabet-weighted costs just 0.1886. On the other hand, the event sequences $\mathcal{S}_3$ and $\mathcal{S}_5$ have an edit distance 0.1111 with the unit costs, but with the alphabet-weighted costs their distance is as long as 0.5909. In the same way, two pairs of sequences that have the same edit distance with one type of operation costs need not to have the same edit distances when some other type of operation costs are used. For example, with the unit costs the edit distance between the event sequences $\mathcal{S}_3$ and $\mathcal{S}_6$ in Figure 4.6 is 0.3389, as is the edit distance between the event sequences $\mathcal{S}_2$ and $\mathcal{S}_5$ in the same set of sequences. However, with the alphabet-weighted costs the edit distances between these pairs of event sequences are 0.6386 and 0.1886, respectively. Similar phenomena were also observed with all the other sets of alarm sequences.

With the sets of WWW page request sequences, our conclusions were also similar to the conclusions with the sets of alarm sequences. Figure 4.12 presents how the edit distances between event sequences preceding requests for the four WWW pages using different types of operation costs are related to each other. In these sets of event sequences there are typically many pairs of sequences that are either completely similar or completely dissimilar with both types of operation costs; only in the case of the KDD link page there are no completely similar pairs of sequences, even though the edit distances with both types of operation costs agree on the most similar pair of sequences. Also the plot of the edit distances between the sequences preceding the "What is Linux" page is different from the other plots. The explanation to this difference is simple: the number of sequences preceding the requests for this page is higher than the numbers of sequences in the other sets (see Table 4.3), and therefore, this set of sequences is also more heterogeneous than the other sets. Still, in all these cases of WWW sequences, using different types of operation costs clearly leads to different kinds of similarity notions.

The results of the experiments with event type sequences preceding the requests for the different WWW pages were mainly the same as with the sets of event sequences. However, with the "What is Linux" page the results of these experiments with event type sequences differ from the results of the experiments with event sequences, as can be seen in Figure 4.13. This difference is due to the nature of the sequences in these sets. In the set of event type sequences there are several pairs of sequences that have exactly the same edit distance when the unit costs are used, but with the alphabet-weighted costs their edit distances are very different. In the corresponding event sequences, the occurrence times of the common events of these sequences are different. This means that the lengths of `Move-`

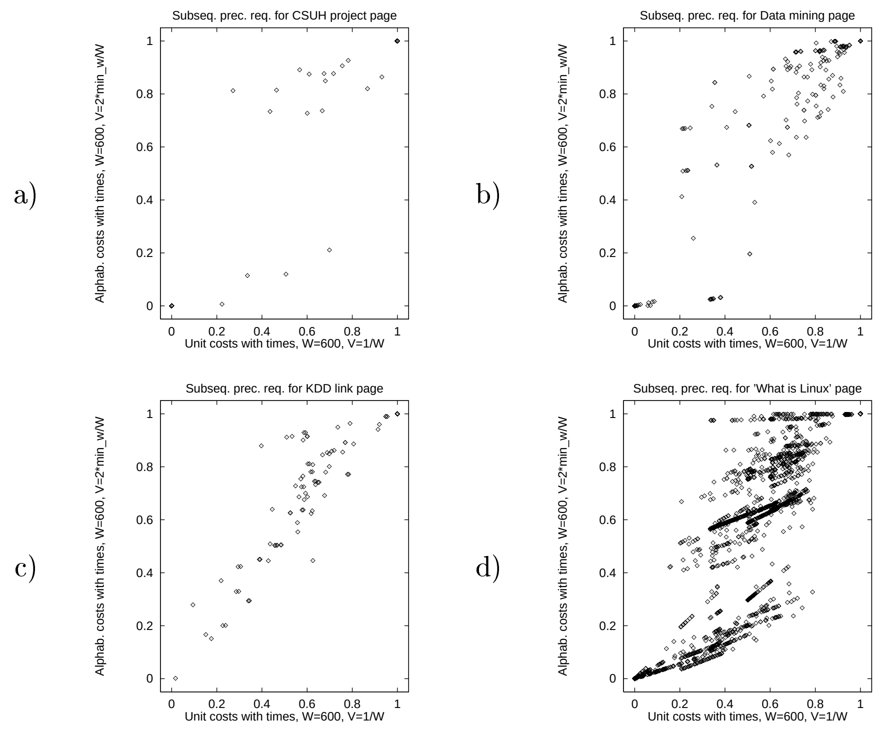


Figure 4.12: Comparison of edit distances between event sequences preceding requests of the a) CSUH project, b) Data mining, c) KDD link, and d) "What is Linux" page with the window width $W = 600$ using the unit costs with $V = \frac{1}{W}$ and the alphabet-weighted costs with $V = \frac{2 \cdot \mathsf{min\_w}}{W}$.

operations vary much, and thus, the probability that two pairs of event sequences have exactly the same edit distance is smaller than in the case of event type sequences, even with the unit costs.

Our experiments show that by using different types of costs for edit operations we can obtain distance measures that describe the set of sequences from different viewpoints. Using some other set of alphabet-weighted operation costs could also have led to a very different kind of edit distances. Because different types of operation costs may suit different situation and different applications, it is not, unfortunately, possible to give any general, application independent rules, how the set of operation costs should be chosen.

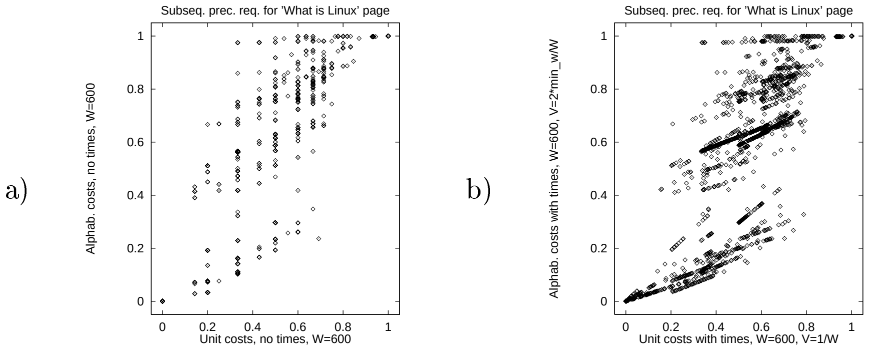


Figure 4.13: Comparison of edit distances between a) event type sequences and b) event sequences preceding requests of the "What is Linux" page with the window width $W = 600$ using the unit costs with $V = \frac{1}{W}$ and the alphabet-weighted costs with $V = \frac{2 \cdot \mathsf{min\_w}}{W}$.

Effect of the window width W

Let us now study how changing the window width W influences edit distances between sequences. For each of the chosen alarm types and WWW pages we used three window widths in extracting subsequences preceding their occurrences. In the telecommunication alarm data we used window widths of 30, 60 and 120 seconds, and in the WWW page request log window widths of 60, 300 and 600 seconds. For the parameter V we used the basic value $\frac{1}{W}$ with the unit costs and the basic value $\frac{2 \cdot \mathsf{min\_w}}{W}$ with the alphabet-weighted costs.

When the window width changes, the sequences themselves also change. Therefore, the first problem to be solved was, whether it is reasonable to compare the edit distances obtained from such sets of sequences at all.

When a small window width is used, the sequence preceding a certain type of an event is typically rather short, or even an empty sequence. When we use a greater window width, the sequence is usually longer, but it is also possible that the length of the sequence does not change at all. The way the sequence changes is highly dependent on the data set as well as on the type of the event that the sequence precedes. Note that also the number of empty sequences often alters when the window width changes; see Tables 4.2 and 4.3 for the exact numbers. In addition, given two window widths W_1 and W_2, where $W_1 < W_2$, a sequence preceding an event of a certain type extracted using the window width W_1 is always the end part of the corresponding sequence extracted using the window width W_2. This means that the sets of sequences preceding events of the given type with different window widths are connected to each other, and thus, comparing their edit distances is justified.

Figure 4.14 presents edit distances between event sequences preceding alarms of the type 1400 using both the unit and the alphabet-weighted operation cost with different window widths. The plots in Figures 4.14a and 4.14c compare the edit distances $d_{\mathcal{S}}$ between event sequences with the window widths $W = 30$ and $W = 60$, and the plots in Figures 4.14b and 4.14d the edit distances $d_{\mathcal{S}}$ between event sequences with the window widths $W = 30$ and $W = 120$. The differences between the plots a and b (c and d) indicate that the window width really influences the edit distances. The differences between the plots a and c with respect to the plots b and d, however, are due to the different operation costs used. The results of the experiments with event type sequences were very similar.

We also compared the edit distances between sequences preceding alarms of the other interesting alarm types. The general conclusions about the influence of the set of operation costs, and especially the influence of the window width W also hold good for each of these test cases. Moreover, the results of the experiments with the sets of event type sequences are similar to these results with these sets of the event sequences.

We also made the same kind of experiments with the WWW log data. Figure 4.15 shows comparisons of edit distances $d_{\mathcal{S}}$ between event sequences preceding requests for the KDD link page with the chosen window widths W. The plots in Figures 4.15a and 4.15c compare the edit distances $d_{\mathcal{S}}$ between event sequences with the window widths $W = 60$ and $W = 300$, and the plots in Figures 4.15b and 4.15d the edit distances $d_{\mathcal{S}}$ between event sequences with the window width $W = 60$ and $W = 600$. The conclusion that we can make from these plots are similar to the case of sets of alarm sequences, namely, that the window width W truly influences the edit distances. Note that the relationship between the edit distances with the window widths $W = 300$ and $W = 600$ is nearly linear (Figures 4.15b and 4.15d). The results of experiments with the corresponding event type sequences are similar to these results.

When we compared the edit distances between sequences preceding requests for the other chosen WWW pages, we found out, similarly to the case of sets of alarm sequences, that the edit distances in each case are very dependent on the set of the sequences we are studying. In general, the results of the experiments with the different window widths W and with the different types of operation costs vary. Unlike with the sets of alarm sequences the results of experiments with the sets of event type sequences

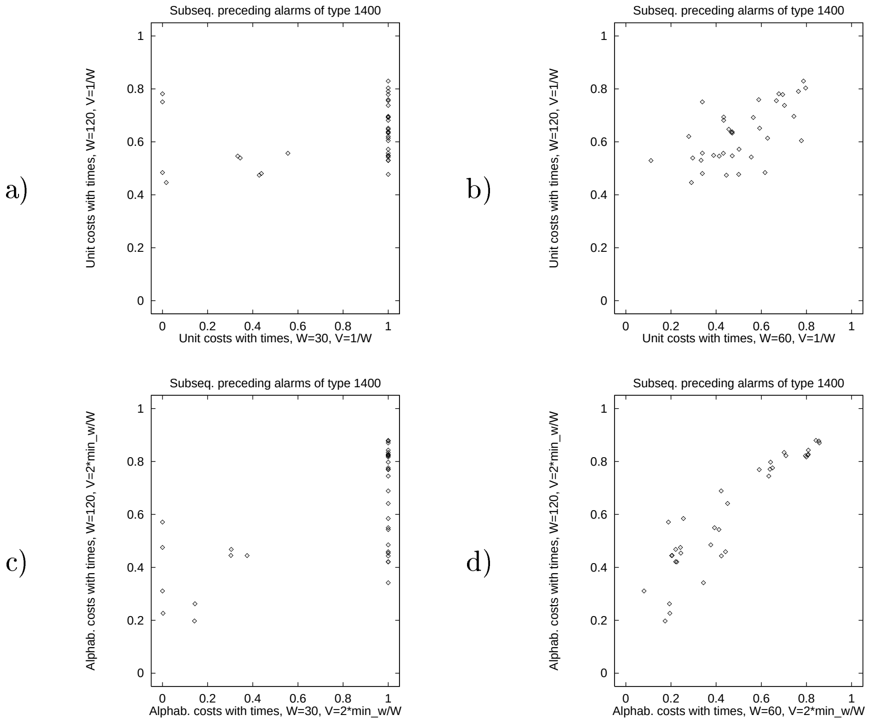


Figure 4.14: Comparison of edit distances between event sequences preceding alarms of the type 1400 when the window width W is varied using (a-b) the unit and (c-d) the alphabet-weighted operation costs.

and with the sets of event sequences were not always very similar, especially with the unit operation costs. Particularly, with the sets of sequences preceding requests for the "What is Linux" pages the results of these experiments differ quite remarkably from each other. This is due to the fact that the sequences change a great deal when the window width increases, and even if the event type sequences with the different window widths are similar, the event sequences contain many long moves, leading to greater differences between the edit distances.

What then actually happens to the edit distances when the window width W changes? Let us start by studying a case where two sequences do not change at all, while the window width alters. Consider for example the set of sequences preceding requests for the KDD link page given in Figure 4.7. The sequences $\mathcal{S}_4$ and $\mathcal{S}_6$ are the same with every chosen window width W. Therefore, the corresponding event type sequences S_4 and S_6 are also always the same. The edit distance between these event type

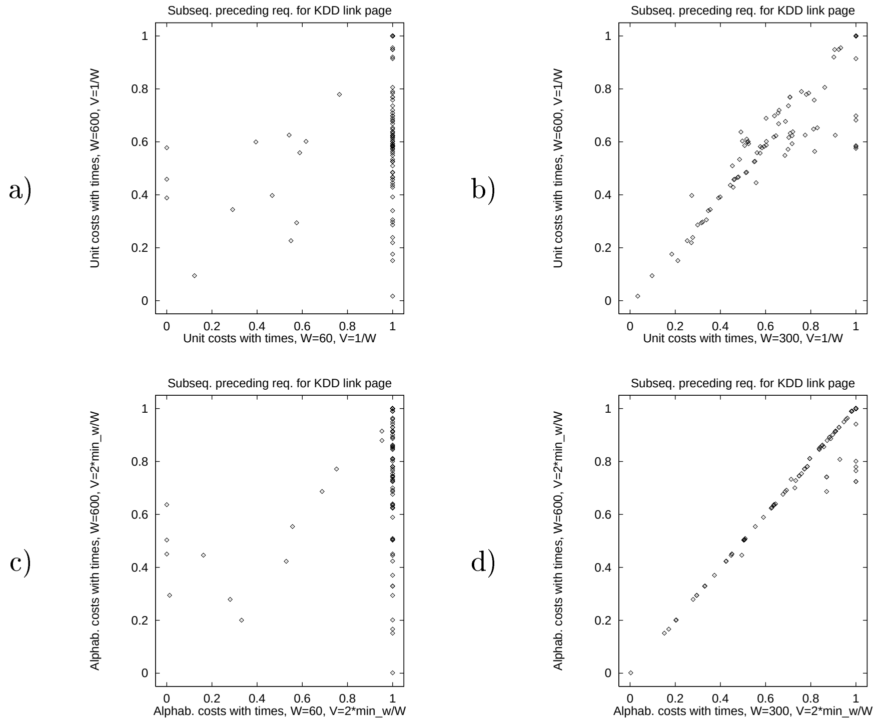


Figure 4.15: Comparison of edit distances between event sequences preceding requests for the KDD link page when the window width W is varied using (a-b) the unit and (c-d) the alphabet-weighted operation costs.

sequences is 0.5556 with the unit costs, and 0.5540 with the alphabet-weighted costs. The edit distances between event sequences, however, vary with the different window widths. This can be explained by the fact that when the window width W increases, V decreases, and therefore, the costs of moving events is also lower. This means that, with greater window widths, the edit distance between the event sequences is shorter than with smaller window widths. This conclusion is true with both the unit and the alphabet-weighted operation costs. Similar pairs of sequences were also found in our other test sets.

Consider now two sequences, either event sequences or event type sequences, that change when the window width is altered. Assume that we know the edit distance between these sequences with some small window width. When the window width increases, the edit distance can either stay the same, or it can become shorter or longer, depending on how the sequences change, e.g., how much their lengths increase and what types of events are added to the sequences. Because it is impossible to predict what the actual changes in each case will be like, we cannot give any general rules on how the edit distances change, when the window width changes. The results of our experiments still show that the edit distances between the sequences change, and that the order of the edit distances between different pairs of sequences is also altered, when the window width changes.

Effect of the parameter V

In the first set of our experiments, we compared the edit distances d_S between event type sequences to the edit distances $d_{\mathcal{S}}$ between event sequences. In those experiments we used the parameter value $V = \frac{1}{W}$ with the unit costs, and the parameter value $V = \frac{2 \cdot \mathsf{min\_w}}{W}$ with the alphabet-weighted costs. This means that moving an event was always preferred to deleting an event first and then inserting one, i.e., all `Move`-operations, regardless of the length of the move, were considered to be cost-effective. The results of the experiments showed that in such cases the edit distances d_S and $d_{\mathcal{S}}$ are in most cases positively linearly correlated.

The parameter V influences the cost of `Move`-operations. This means that the optimal operation sequence can also depend on V, and therefore, changing the value of V can also alter the edit distance $d_{\mathcal{S}}$ between event sequences. In order to find out what this change in edit distances $d_{\mathcal{S}}$ is really like, we made some new experiments using the sets of alarm sequences with the window width $W = 60$, as well as the sets of WWW page requests with the window width $W = 600$. The values of V used in these experiments are given in Tables 4.4 and 4.5. With the smallest values of V, i.e., the values $\frac{1}{W}$ and $\frac{2}{W}$ with the unit costs, or the value $\frac{2 \cdot \mathsf{min\_w}}{W}$ with the alphabet-weighted costs, all `Move`-operations are cost-effective, regardless of the length of the move. However, with all the other values of V the longest moves are not cost-effective. If we use the value $V = \frac{4}{W}$ with the unit costs, for example, any type of an event can be moved at maximum by a half of the window width W used. In the case of the alphabet-weighted costs, the situation is more complicated, because the length of a cost-effective move depends both on the value of V and the type of the event (see Section 4.2.2).

Figure 4.16 describes how edit distances d_S between event type sequences and edit distances $d_{\mathcal{S}}$ between event sequences preceding alarms of the type 1400 in the telecommunication alarm data are related to each other, when the unit operation costs are used. The edit distances $d_{\mathcal{S}}$ in Figure 4.16a are computed using the parameter value $V = \frac{8}{W}$, and in Figure 4.16b using the parameter value $V = \frac{120}{W}$. The relationships between

a) b)

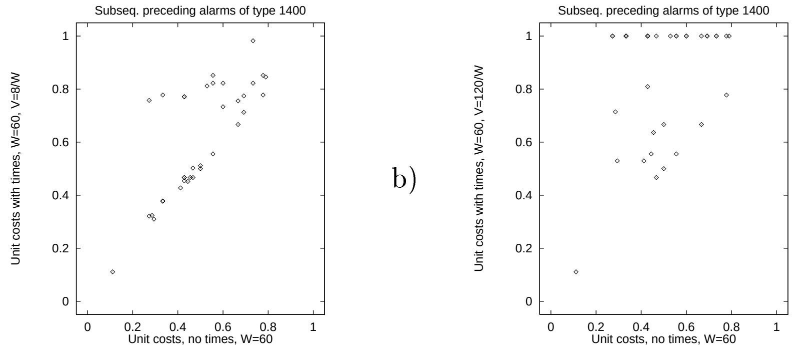


Figure 4.16: Comparison of edit distances between event type sequences and event sequences preceding alarms of the type 1400 within 60 seconds using the unit costs with a) $V = \frac{8}{W}$ and b) $V = \frac{120}{W}$.

the edit distances d_S and $d_{\mathcal{S}}$ are not linear, and the same conclusions holds good with all the other values of V that restrict the lengths of moves. The absolute values of the edit distances $d_{\mathcal{S}}$ typically increase, when the value of the parameter V increases. Especially with the value $V = \frac{120}{W}$, when the maximum length of a move is one second, there are many pairs of sequences that are completely dissimilar, i.e., their edit distance has the value one.

The results of these experiments with the sets of alarm sequences also clearly indicate that the order of the edit distances $d_{\mathcal{S}}$ changes when different values of V are used. Consider, for example, the event sequences $\mathcal{S}_5$, $\mathcal{S}_6$, and $\mathcal{S}_9$ in the set of alarm sequences in Figure 4.6. With the parameter value $V = \frac{8}{W}$, their edit distances are $d_{\mathcal{S}}(\mathcal{S}_5, \mathcal{S}_6) = 0.3212$ and $d_{\mathcal{S}}(\mathcal{S}_6, \mathcal{S}_8) = 0.3238$, but with the parameter value $\frac{120}{W}$ their edit distances are $d_{\mathcal{S}}(\mathcal{S}_5, \mathcal{S}_6) = 1.0000$ and $d_{\mathcal{S}}(\mathcal{S}_6, \mathcal{S}_8) = 0.7143$. And further, as Figure 4.16 shows, the orders of the edit distances d_S and $d_{\mathcal{S}}$ are different.

Similarly, Figure 4.17 describes how edit distances d_S between event type sequences that precede requests for the KDD link page are related to edit distances $d_{\mathcal{S}}$ between the corresponding event sequences, with the unit operation costs. The edit distances $d_{\mathcal{S}}$ in Figure 4.17a are computed using the value $V = \frac{24}{W}$, and in Figure 4.17b the value $V = \frac{120}{W}$. In these cases, as well as with the other sets of WWW page request sequences, the relationships between the edit distances d_S and $d_{\mathcal{S}}$ are not linear, when all `Move`-operations are not cost-effective. Moreover, the orders of the edit distances d_S and $d_{\mathcal{S}}$ as well as the orders of the edit distances $d_{\mathcal{S}}$ with different values of V vary in these sets of WWW page request sequences.

The same kind of experiments were done using edit distances with the alphabet-weighted costs. Figure 4.18 describes how edit distances d_S be-

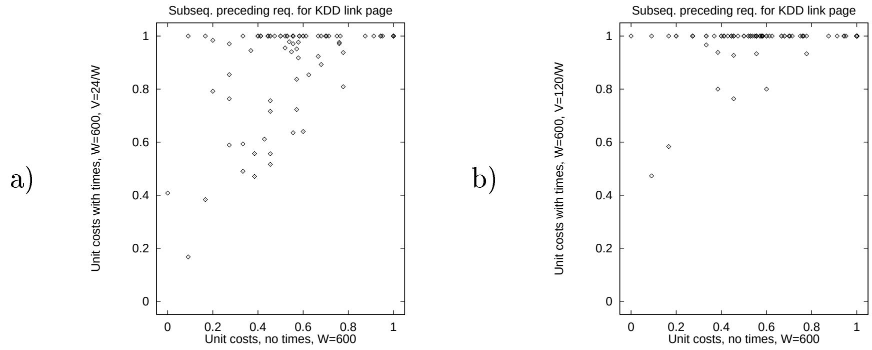


Figure 4.17: Comparison of edit distances between event type sequences and event sequences preceding requests for the KDD link page within 600 seconds using the unit costs with a) $V = \frac{24}{W}$ and b) $V = \frac{120}{W}$.

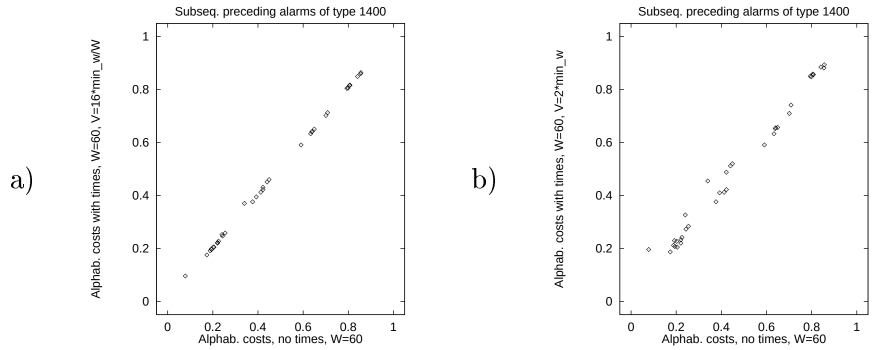
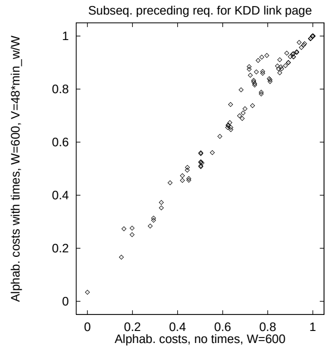


Figure 4.18: Comparison of edit distances between event type sequences and event sequences preceding alarms of the type 1400 within 60 seconds using the alphabet-weighted costs with a) $V = \frac{16 \cdot \mathsf{min\_w}}{W}$ and b) $V = 2 \cdot \mathsf{min\_w}$.

tween event type sequences and edit distances $d_{\mathcal{S}}$ between event sequences preceding alarms of the type 1400 in this case are related to each other. The edit distances $d_{\mathcal{S}}$ in Figure 4.18a were computed using the value $V = \frac{16 \cdot \mathsf{min\_w}}{W}$, and in Figure 4.18b the value $V = 2 \cdot \mathsf{min\_w}$. Similar plots for the sets of sequences preceding requests for the KDD link page are given in Figure 4.19. The parameter values used in computing the edit distances $d_{\mathcal{S}}$ in these two plots are $V = \frac{48 \cdot \mathsf{min\_w}}{W}$ (Figure 4.19a), and $V = 2 \cdot \mathsf{min\_w}$ (Figure 4.19b).

Earlier, in our first set of experiments on sequence similarity we noticed that, with the alphabet-weighted operation costs, the correlation between edit distances $d_{\mathcal{S}}$ and $d_{\mathcal{S}}$ is even more clearly linear than with the unit oper-

a) 

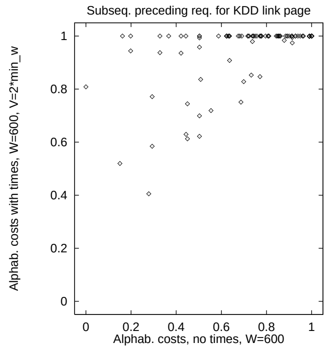
b) 

Figure 4.19: Comparison of edit distances between event type sequences and event sequences preceding requests for the KDD link page within 600 seconds using the alphabet-weighted costs with a) $V = \frac{48 \cdot \mathsf{min\_w}}{W}$ and b) $V = 2 \cdot \mathsf{min\_w}$.

ation costs, presuming that all moves of events are cost-effective, regardless of their lengths; see Figure 4.8b and 4.9b. Still, we assumed that when all moves are not cost-effective, the relationship between edit distances between event type sequences and event sequences would not be linear any more, similar to the case of the unit operation costs. This assumption, however, turned out to be wrong, especially with the sets of sequences from the telecommunication alarm data. Namely, the edit distances $d_{\mathcal{S}}$ between event sequences preceding the chosen type of alarms turned out to be also clearly linearly correlated with the edit distances d_S between event type sequences with the highest values of V, when the alphabet-weighted operation costs were used. This means that the orders of the edit distances were also generally the same. The only exception to this were the sets of sequences preceding alarms of the type 7712. In this case, the edit distances d_S and $d_{\mathcal{S}}$ differ with the parameter value $V = \frac{16}{W}$ already, and differ even more with all the higher values of V. The order of the edit distances also changed in all these cases. These differences in the behavior with this alarm type perhaps depend on that the number of occurrences of the alarm type 7712 is quite high (see Table 4.2), and that the set of the sequences preceding these occurrences is quite heterogeneous. Thus, the large variation in the edit distances is only natural. Another reason for the wide distribution of the edit distances is that the variation in the lengths of the moves is larger than in the other three cases of alarm sequences.

With the sets of sequences preceding requests for the chosen WWW pages, the edit distances d_S and $d_{\mathcal{S}}$ were also linearly correlated with the small values of V. However, a certain kind of a threshold value for these cases was typically $V = \frac{48}{W}$. With this value and all the higher values of V, the relationship between the edit distances $d_{\mathcal{S}}$ and $d_{\mathcal{S}}$ was no longer linear. Moreover, these results mean that with small values of V, the order of the edit distances is mainly the same, but when the higher values of V are used, the order also alters. In the sets of sequences from the WWW log data there was also one exception. The edit distances $d_{\mathcal{S}}$ between sequences preceding requests for the "What is Linux" page started to differ from the edit distances $d_{\mathcal{S}}$ with the smallest parameter value $V = \frac{2 \cdot \mathsf{min\_w}}{W}$ already, as we saw in the first type of experiments with sequence similarity. The explanation to this phenomenon is the same as with the sets of sequences preceding alarms of the type 7712 above. That is, the number of sequences in these sets is high (see Table 4.3), and thus, the distribution of the edit distances becomes wide. And even if the sequences have a long common event type subsequence, the differences in the occurrence times, and therefore, also in the lengths of the moves vary a lot, leading to greater differences between the edit distances.

Why do the edit distances with the alphabet-weighted costs then differ from the edit distances with the unit costs? One reason for this probably lies in the values of the parameter V used with the alphabet-weighted costs; all the values used are multiples of the value $2 \cdot \mathsf{min\_w}$. In the telecommunication alarm data one alarm type occurs 12 186 times in the whole alarm sequence, and thus, the value of the constant $\mathsf{min\_w}$ in this case is extremely low. This in turn means that all the values of V are also very low, and even if we use the maximum possible value $V = 2 \cdot \mathsf{min\_w}$, only the lengths of the moves of the 39 most common alarm types are restricted, when there are a total of 287 alarm types and the window width W used is 60 seconds. This explains why changing the value of V cannot make any remarkable difference in the edit distances between alarm sequences.

In the WWW log data the situation with the alphabet-weighted costs was slightly closer to the case of the unit costs. This can also be explained with the value of the constant $\mathsf{min\_w}$: in this case the highest number of references to one WWW page was 1 848, which means that the value of $\mathsf{min\_w}$ was rather low, but not at all as low as in the case of the telecommunication alarm data. When we used the window width W of 600 seconds, this in practice meant that for the 1 802 most common WWW pages out of 6 023 pages at least some moves were not cost-effective. In other words, those pages that are referred to more than three times could not be moved freely. If we would have liked to restrict the lengths of moves of several event types in either of the data sets, we should have either eliminated the most common types of events from the sequence considered, or violated the restriction that $V \leq 2 \cdot w(e)$ for all $e \in \mathcal{E}$. How such changes would influence the edit distances is left for future study.

As stated already, the results of these experiments with the various test sets were mainly the same. There are, however, some cases where the value of the parameter V is known to have only a small effect on edit distances. For the first, assume that two sequences have exactly the same subsequence, i.e., no events in this subsequence need to be moved, and in addition to that they have only a few non-common events. Then the edit distance between these sequences is always constant with respect to the value of V. An example of such a sequence pair is the pair of the event sequences $\mathcal{S}_3$ and $\mathcal{S}_5$ given in Figure 4.6. These sequences differ only by the alarm $(1553, 8)$, and therefore, their edit distance $d_{\mathcal{S}}(\mathcal{S}_3, \mathcal{S}_5)$ with the unit costs is always 0.1111 and with the alphabet-weighted costs 0.5909. On the other hand, an edit distance between two sequences can always have a high value. In such a case, the sequences do not have many common events and/or the moves needed are very long. An example of such a sequence pair is the pair of the event sequences $\mathcal{S}_1$ and $\mathcal{S}_9$ given in Figure 4.6. In neither case does the value of V have any particular influence on the edit distances.

Hierarchies of sequences of events

The edit distances between sequences give us important information about how similar the sequences are. The edit distances can also be used to build hierarchies of sequences. Such hierarchies are needed, for example, if we want to make a query to a collection of event sequences and efficiently find all sequences similar to the query. Methods for building such indexes for time series and other numerical sequences have been proposed, for example, in [AFS93, BYÖ97, BÖ97, FRM93]. One way of constructing such an index in the case of sequences of events is to cluster the sequences with respect to their edit distances.

On the other hand, an interesting problem in analyzing sequential data is to predict an occurrence of an event of a given type. This problem can also be considered as finding explanations for what usually happens before an event of a given type occurs. In telecommunication network monitoring, for example, we could use such information to detect the probable occurrence of a severe fault early enough to prevent it, or at least to be prepared to fix its consequences. We could try to solve this prediction problem, for example, by searching from the sequence all *episode rules* [MTV95, MTV97, Toi96] 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 to this problem would be to search for typical situations that precede occurrences of the given event type. Such situations can be found, for example, by clustering the sequences preceding the occurrences of the events of the given type within a given time period.

Such a hierarchy of sequences can be constructed, for example, using the standard agglomerative hierarchical clustering. Similarly to the case of binary attributes (Section 3.5), we tested building clustering trees of sequences using the three hierarchical clustering methods presented in Appendix A, namely, the single, the complete, and the average linkage methods. In the following we present some results on clustering of event sequences. In most cases the clusterings of event type sequences are the same, or at least very similar to the clusterings of event sequences.

The sequences of events in our experiments were often very long. Thus, when we draw a clustering tree, it is not possible to give the actual sequences in these trees. Instead, we represent the sequences by symbols like $\mathcal{S}_i$ where i indicates the ordinal of the sequence when the set of sequences are given as a list of sequences like in Figures 4.6 and 4.7. However, the clustering trees where the sequences are represented with the symbols $\mathcal{S}_i$ as such are not very informative to the reader. In order to be able to really evaluate the naturalness and the goodness of the clustering trees the reader must be able to see the actual sequences at the same time. Therefore, we present here just a few examples of clusterings of event sequences in the set of alarm sequences preceding occurrences of the alarm type 1400 with the window width $W = 60$, and in the set of WWW page request sequences preceding requests for the KDD link page with the window width $W = 600$.

Earlier in this section we found that the edit distances with the unit and the alphabet-weighted costs can differ a lot. This leads to the assumption that the clustering trees resulting from the edit distances with these costs should also be different. This assumption was confirmed by our experiments. To name an example, in Figure 4.20 we have two clustering trees of the event sequences preceding alarms of the type 1400 using the window width $W = 60$ (see Figure 4.6), and the unit costs with $V = \frac{1}{W}$ (Figure 4.20a) and the alphabet-weighted costs with $V = \frac{2 \cdot \mathsf{min\_w}}{W}$ (Figure 4.20b). Both the clustering trees were produced with the single linkage clustering method. These clustering trees really are dissimilar, because the edit distances on which the clustering of the sequences was based were computed with different operation costs. With other sets of sequences and other clustering methods, the results were typically similar. This tells us that the choice of the type of the operation costs have quite an influence on whether two sequences are grouped together or not. Note that the clustering tree

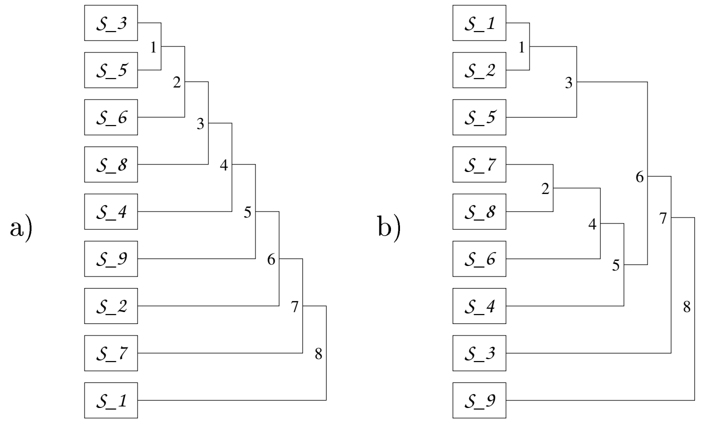


Figure 4.20: Clustering trees of event sequences preceding alarms of the type 1400 produced with the single linkage clustering method when the window width is 60 seconds and a) the unit costs with $V = \frac{1}{W}$, and b) the alphabet-weighted costs with $V = \frac{2 \cdot \mathsf{min\_w}}{W}$ are used.

produced with the single linkage method by using the edit distances with the unit costs is a good example of the chaining of clusters.

Clustering experiments were also made with the sets of WWW page request sequences. In Figure 4.21 there are two clustering trees of sequences preceding requests for the KDD link page with the window width $W = 600$. The clustering trees were produced with the single linkage method, and the edit distances between event sequences were computed using the unit costs with $V = \frac{1}{W}$ (Figure 4.21a), and the alphabet-weighted costs with $V = \frac{2 \cdot \mathsf{min\_w}}{W}$ (Figure 4.21b). In this case, the clustering trees produced with the same clustering method, but using edit distances computed with different operation costs, are not exactly similar, although some parts of the trees are alike. Further, the results with the other sets of WWW page request sequences and with the different clustering methods confirm the idea that the choice of the type of operation costs strongly influences the resulting clustering tree.

In Figure 4.22 there are six clustering trees of event sequences preceding alarms of the type 1400 when the window width used is 60 seconds. The clustering trees in the upper row of the figure are based on the edit distances computed using the unit operation costs with $V = \frac{1}{W}$, and in the lower row of the figure on the edit distances computed using the alphabet-weighted operation costs with $V = \frac{2 \cdot \mathsf{min\_w}}{W}$. In producing the trees in Figures 4.22a and 4.22d we used the single linkage method, the trees in Figures 4.22b and 4.22e the complete linkage method, and the trees in Figures 4.22c

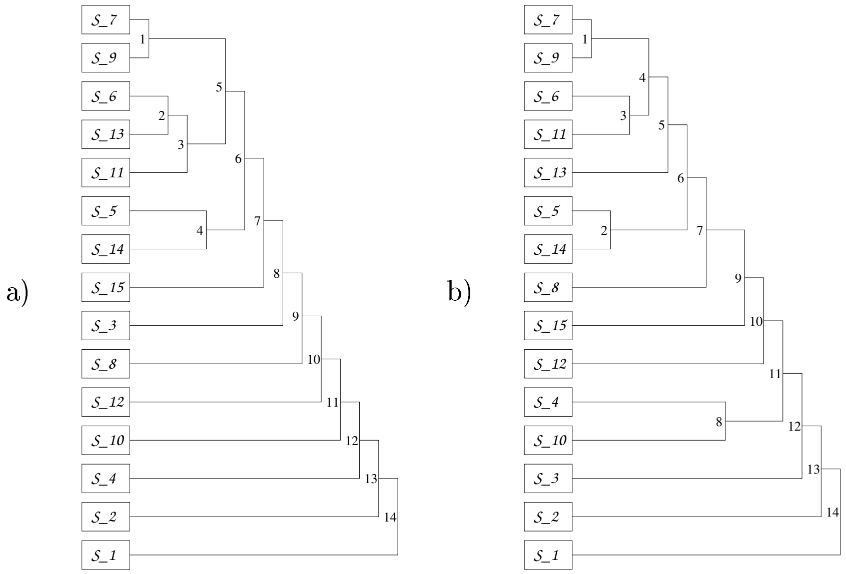


Figure 4.21: Clustering trees of event sequences preceding requests for the KDD link page produced with the single linkage method when the window width is 600 seconds and a) the unit costs with $V = \frac{1}{W}$, and b) the alphabet-weighted costs with $V = \frac{2 \cdot \mathsf{min\_w}}{W}$ are used.

and 4.22f the average linkage method. The clustering trees produced using edit distances computed with different edit operation costs are different, as noted earlier in this section. This conclusion holds good for each of the sets of sequences considered in our experiments. In addition to that, the trees produced with different clustering methods are typically also different, even though the differences in the tree structures may not always be very large. Exceptions to this rule are the clustering trees in Figures 4.22e and 4.22f: they are produced with different methods, but their structures are exactly the same. However, as stated already with the clustering of binary attributes, the more typical situation is that clustering trees based on the same set of edit distances, but produced with different clustering methods, are dissimilar to each other.

Above we presented clustering trees produced with different clustering methods and using edit distances computed with different edit operation costs. In these cases, however, we considered only edit distances computed using the chosen basic values of the window width W and the parameter V. Because our experiments with the different values of these parameters earlier showed that the edit distances are greatly influenced by these values,

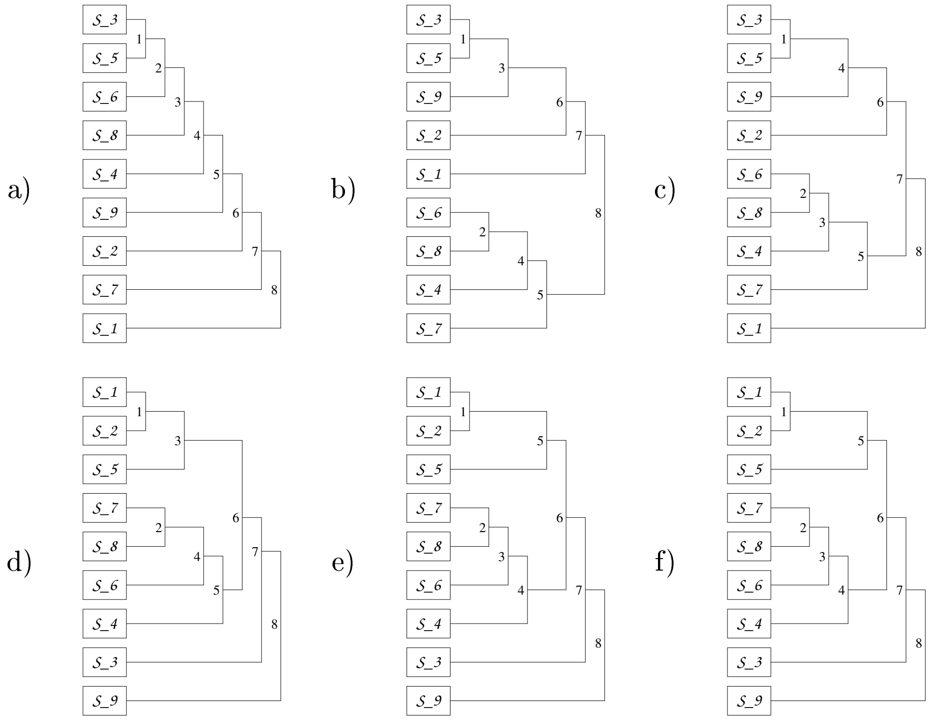


Figure 4.22: Clustering trees of event sequences preceding alarms of the type 1400 produced with the single (left column), complete (center column), and average (right column) linkage methods, when the window width is 60 seconds, and the unit costs with $V = \frac{1}{W}$ (upper row) and the alphabet-weighted costs with $V = \frac{2 \cdot \mathsf{min\_w}}{W}$ (lower row) are used.

it should be intuitively clear that also the clustering trees based on these various edit distances would be different from those presented here.

4.5 Discussion

In this chapter we have studied how similarities between sequences of events can be determined. Our approach to this problem is based on computing edit distances between sequences. To define an edit distance between two sequences, we have to find the sequence of edit operations that transforms one sequence into another and has the minimum cost. We defined such an edit distance between both sequences of events with and without occurrence times.

The results of our experiments on edit distances between both event type sequences and event sequences using real-life data sets showed that using this kind of distance measures we can find similar sequences. In addition to that, the choice of the parameter values needed in computing the edit distances turned out to be very important. In the real-life data sets, the window width W, and especially the type of the edit operation costs had a remarkable effect on what kind of sequences were considered to be similar or dissimilar. With the sets of sequences from the WWW page request log, the parameter V needed in computing costs of moving events also had a great influence on the edit distances. On the other hand, with the sets of sequences from the telecommunication alarm data the parameter V did not have that much influence on the edit distances.

As our experimental results indicate, with different choices of the parameter values we can achieve very different notions of event sequence similarity. What these experiments do not give us, unfortunately, is a general rule how to choose the values of the parameter for each situation. Therefore, we should have enough domain knowledge to be able to select the right parameter values for each application area, or we have to find such values experimentally.

Chapter 5

Similarity between event types in sequences

In the previous chapter we considered how similarity between sequences of events could be defined. Another interesting question concerning sequential data is how to define a useful notion of similarity between event types occurring in event or event type sequences. Such a similarity notion can provide us with important information about the relationships between event types, for example. Our main idea in defining similarity between event types in sequences is that two event types are similar if they occur in similar contexts in the sequences. Therefore, we first study different ways of defining a context of an occurrence of an event type, and then, how similarity between two event types is deemed based on similarity between the sets of contexts of all their occurrences. We also present some experimental results with different measures of event type similarity, and show how these similarities can be used to build hierarchies of event types.

We start this chapter by giving some definitions on event types in Section 5.1. Then in Section 5.2 we consider ways of defining the context of an occurrence of an event type, and in Section 5.3 we present ways of computing similarity between sets of contexts. After that, in Section 5.4 we give algorithms for computing these similarities between event types. Experimental results on event type similarity are represented in Section 5.5. Parts of the work described in this chapter are presented in [MM99].

5.1 Event types in sequences

In this chapter we consider the same model of event sequences (and event type sequences) as in Chapter 4. But instead of the sequences themselves, we now focus on the types of events occurring in the sequences.

The set $\mathcal{E}$ of all possible event types depends on the domain area, and even on the application considered. In telecommunication network monitoring, for example, a log of alarms occurring in the network can been seen as an event sequence. Here the set of possible event types is formed by the types of the alarms that network elements can send. A log of WWW page requests from the single session of a user can also be viewed as an event sequence. In this case, the WWW pages are the possible event types. Yet another example of real-life event type sequences are protein sequences studied in molecular biology. Each protein sequence is a chain of amino acids, and therefore, in this setting, the set of possible event types consists of the twenty amino acids listed in Table 5.1. Note that these twenty are the amino acids typically found in the protein sequences; exceptionally a few non-standard amino acids might be present in the sequences [SM97].

When we look at a particular event sequence (or event type sequence), there seldom occur events of every type in the set $\mathcal{E}$. Moreover, if we consider different sequences over the same set $\mathcal{E}$, the sets of event types occurring in these sequences typically vary a lot. Thus, we have the following definition of a subset of event types.

Definition 5.1 Let $\mathcal{E}$ be a set of all possible event types. The number of event types in this set, i.e., the size of the set is denoted by $|\mathcal{E}|$. Then assume that $\mathcal{S}$ is an event sequence over $\mathcal{E}$. The set of types of events occurring in the sequence $\mathcal{S}$ is defined as

$$\mathcal{E}_{\mathcal{S}} = \{ e \mid \exists (e_i, t_i) \in \mathcal{S} \text{ so that } e_i = e \}.$$

This set $\mathcal{E}_{\mathcal{S}}$ is a subset of the set $\mathcal{E}$. The number of event types in the set $\mathcal{E}_{\mathcal{S}}$ is the size of the set, and it is denoted by $|\mathcal{E}_{\mathcal{S}}|$. Similarly, a set of types of events occurring in an event type sequence S over $\mathcal{E}$ is defined as

$$\mathcal{E}_S = \{ e \mid \exists e_i \in S \text{ so that } e_i = e \}.$$

This set $\mathcal{E}_S$ is also a subset of $\mathcal{E}$ with a size $|\mathcal{E}_S|$. □

In examples and experiments of this chapter we use both artificial and real-life data. The real-life data sets used in examples are a *telecommunication alarm sequence* and a set of *protein sequences.*

Example 5.1 Consider the example event sequence $\mathcal{S}$ in Figure 5.1 (the same sequence as in Figure 4.1). The set of event types occurring in this sequence is $\mathcal{E}_{\mathcal{S}} = \{A, B, C, D, E, F\}$. □

	One-letter code	Three-letter code	Name
1	A	Ala	Alanine
2	C	Cys	Cysteine
3	D	Asp	Aspartic Acid
4	E	Glu	Glutamic Acid
5	F	Phe	Phenylalanine
6	G	Gly	Glycine
7	H	His	Histidine
8	I	Ile	Isoleucine
9	K	Lys	Lysine
10	L	Leu	Leucine
11	M	Met	Methionine
12	N	Asn	Asparagine
13	P	Pro	Proline
14	Q	Gln	Glutamine
15	R	Arg	Arginine
16	S	Ser	Serine
17	T	Thr	Threonine
18	V	Val	Valine
19	W	Trp	Tryptophan
20	Y	Tyr	Tyrosine

Table 5.1: The twenty amino acids commonly found in proteins [SM97].

Example 5.2 The telecommunication alarm sequence $\mathcal{S}_{alarm}$ given in Figure 5.2 (the same sequence as in Figure 4.2) consists of 23 alarms. Types of all these alarms belong to the set

$$\mathcal{E}_{\mathcal{S}_{alarm}} = \{7002, 7010, 7030, 7127, 7172, 7177, 7201, 7311, 7312, 7401\}.$$

Note that this set of ten alarm types is only a small set of all possible alarm types. Typically, in alarm sequences there are hundreds, or even thousands of different types of alarms. □

Example 5.3 The SWISS-PROT protein sequence database [SWI99] contains information about 78 350 protein sequences. One of the protein sequences in this set is the sequence

$$VLSAA\ DKGHV\ KGIWG\ KVGGH\ AGEYA\ AEGLE\ R$$

which is a fragment of the hemoglobin alpha chain of a *Tammar wallaby* (Macropus eugenii).

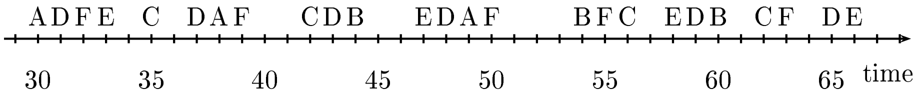


Figure 5.1: An event sequence on the time axis.

Protein sequences can also be seen as event type sequences or event sequences; we can think of amino acids as event types and occurrences of amino acids as actual events. Presenting a protein sequence as an event type sequence then is straightforward. To name an example, an event type sequence corresponding to the protein sequence above is a sequence

$$S_{protein} = \langle\, V, L, S, A, A, D, \ldots, L, E, R \,\rangle$$

when we use the one-letter codes of the amino acids.

Viewing protein sequences as event sequences is more complicated because the amino acids in protein sequences have no occurrence times. However, there is a simple solution to this problem: an occurrence time of each event can be defined as the ordinal of the event in the sequence. Using this definition, the protein sequence above would be represented as the event sequence

$$\mathcal{S}_{protein} = \langle\, (V,1), (L,2), (S,3), (A,4), \ldots, (E,30), (R,31) \,\rangle.$$

These constructed occurrence times of events are, however, very artificial, and thus, sequences where the events do not originally have occurrence times are treated only as event type sequences in the following.

Both the event type sequence $S_{protein}$ and the event sequence $\mathcal{S}_{protein}$ consist of 31 occurrences of amino acids. The set of amino acids occurring in these sequences is the set

$$\mathcal{E}_{\mathcal{S}_{protein}} = \mathcal{E}_{S_{protein}} = \{A, D, E, G, H, I, K, L, R, S, V, W, Y\}.$$

of thirteen amino acids. □

In Definition 5.1 we only considered event types occurring in one event sequence or event type sequence. The definition, however, also applies to a set of sequences. For example, given a set S of event sequences, a set of event types occurring in these sequences would be

$$\mathcal{E}_{\mathsf{S}} = \{\, e \mid \exists\, \mathcal{S} \in \mathsf{S} \text{ so that } \exists\, (e_i, t_j) \in \mathcal{S} \text{ and } e_i = e \,\}.$$

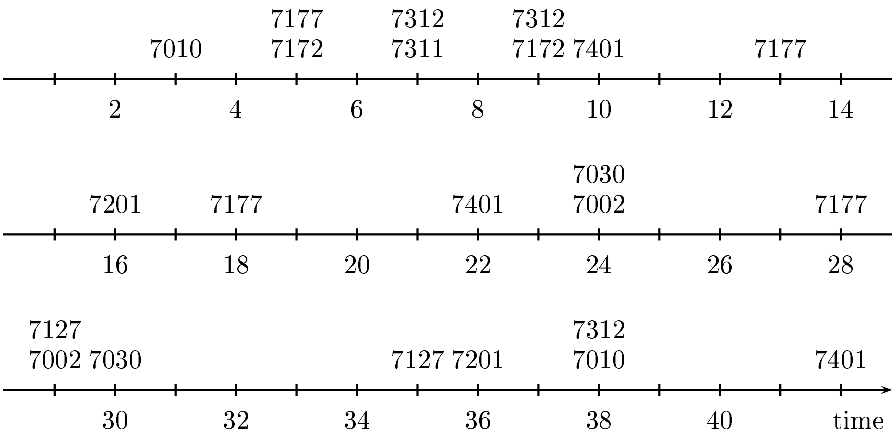


Figure 5.2: An example alarm sequence on the time axis.

Note that in most cases the set $\mathcal{E}_{\mathsf{S}}$ is different from the set $\mathcal{E}_{\mathcal{S}}$ of any single event sequence $\mathcal{S} \in \mathsf{S}$.

When we want to determine similarity between sequences, or make a query to a database of sequences, we can, of course, consider only equality and inequality of event types. In some cases, like when comparing protein sequences [SM97], this is, however, not enough. This means that we need a way of defining similarity between event types occurring in the sequences. Such similarities between event types can, for example, be used in defining the cost of replacing an event with an event of a similar type, when we compute how one sequence of events could be transformed into another (see Section 4.2.4). Information of similarities between event types can also be valuable as such, as it provides insight into the data. In the following we give a general definition of similarity between event types by using a complementary notion of distance.

Definition 5.2 Let $\mathcal{E}$ be a set of event types, and $\mathbf{S}$ a set of all possible event sequences over $\mathcal{E}$. Then a *distance* measure d between event types is defined as $d : \mathcal{E} \times \mathcal{E} \times \mathbf{S} \to \mathbb{R}$. Given two event types $A \in \mathcal{E}$ and $B \in \mathcal{E}$, and a set S of sequences over $\mathcal{E}$ where $\mathsf{S} \in \mathbf{S}$, a distance between the two event types A and B is denoted by $d(A, B; \mathsf{S})$. If there is no risk of confusion, we just write $d(A, B)$. $\square$

The actual distance function d can, of course, be chosen in many different ways. The intuitive idea behind our approach is simple: two event types are similar, if they occur in similar *contexts*. That is, two event types A and B are similar, if the situations in which they occur in sequences resemble each other. Hence, we define similarity between event types A and B by computing the similarity between their sets of contexts.

The idea of using contexts in defining similarities between objects is not new. For example, a method for automatic word sense disambiguation presented in [KE98] is based on word and context similarity measures. In this method words are considered to be similar, if they appear in similar contexts, usually sentences, and contexts are deemed as similar, if they contain similar words. Using contexts in judgments on similarity has also been considered in the behavioral and psychological studies [Tve77]. And in a way, determining similarity between binary attributes using external similarity measures (see Section 3.3) is also based on using contexts of attributes.

Computing similarity between event types using our approach requires us to have answers to two questions:

1. What is the context of an occurrence of an event type?

2. How is similarity between sets of contexts defined?

In the next two sections we discuss alternative answers to these questions. The problem of defining contexts of occurrences of events is studied in Section 5.2, and after that, in Section 5.3 we describe ways of defining similarity between sets of contexts.

In real-life sequences the number of possible event types is typically tens, hundreds, or even thousands. Especially, if there are thousands of event types, computing similarity between all of them can be tedious. On the other hand, there are cases in which we are not even interested in finding similarities between all event types, not even between those occurring in the single sequence, or in the set of sequences considered. This means that we should choose the event types that interest us and compute similarities only between them.

Definition 5.3 Let $\mathcal{E}$ be a set of all possible event types. A set of event types $e \in \mathcal{E}$ whose pairwise similarities we want to compute is called *a set of interesting event types*, and it is denoted by $\mathcal{EI}$. The size of the set $\mathcal{EI}$, i.e., the number of interesting event types, is denoted by $|\mathcal{EI}|$. □

Example 5.4 The types of alarms that can occur in a telecommunication network vary depending on the network management system used. Yet, the set of all possible alarm types can typically consist of thousands of alarm types. If we look at a certain alarm sequence, the number of types of alarms occurring in it can, however, be only some tens or hundreds. But still, in most cases, the size of such a set of alarm types is too large, so that it would be reasonable to consider similarity between all the alarm types within the set.

Each (sub)component of the telecommunication network can produce alarms. Therefore, one possibility for a set $\mathcal{EI}$ are the types of alarms sent by a certain type of subcomponents, or even by a particular subcomponent. As a criterion for alarm types to be accepted into the set $\mathcal{EI}$, we could also use frequencies of the alarm types, i.e., how many occurrences of the alarm types there were in the sequence or in the set of sequences considered. We might, for example, be interested in how similar to each other the alarm types are whose occurrence frequencies are at least hundred. □

Example 5.5 In protein sequences there are usually occurrences of twenty different amino acids (see Table 5.1). In addition to these twenty amino acids, in the sequences of the SWISS-PROT data set [SWI99] there are also occurrences of three non-standard symbols: B, X and Z. Of these the symbol B codes an occurrence of Aspartic acid or Asparagine. On the other hand, the symbol Z means that either Glutamine or Glutamic acid occurs in the sequence, whereas the third non-standard symbol X can correspond to an occurrence of any of the twenty amino acids.

The number of the standard amino acids is a rather reasonable number of event types, and therefore, it is possible to consider all twenty of them as the set $\mathcal{EI}$ of interesting event types. Also the non-standard symbols B, X and Z can be part of the set $\mathcal{EI}$, if measuring similarities between them and the standard amino acids is considered to be relevant. Yet another possibility is to study only a set of some amino acids that have similar biochemical properties; similar size, or similar tendency to bind with water molecules, for example. □

The selection of interesting event types depends on the application and the situation we are considering. A natural requirement would be that the interesting event types should be intuitively similar. But when choosing the interesting event types (similarly to the case of choosing interesting attributes in Chapter 3), it is important to remember that if we consider just the event types known to be associated with each other, some interesting and new relationships between event types may be lost.

5.2 Contexts of events

The first problem to be considered in defining similarity between event types is, how we should define the context of an occurrence of an event type. Usually, the sequence we are studying is too long to be used as a context as such. In the following we introduce a notation that restricts which events of the sequence studied are taken into account when extracting the context from the sequence.

Definition 5.4 Let $\mathcal{E}$ be a set of event types. A *context* of an event (e_i, t_i) in an event sequence $\mathcal{S}$ over $\mathcal{E}$ is denoted by

$$\mathsf{con_f}((e_i, t_i), \mathcal{S}, W),$$

where $W \in \mathbb{R}$ is a time parameter, called a *window width*, that restricts which events are examined in forming the context of the event (e_i, t_i). Similarly, a context of an event e_i in an event type sequence S over $\mathcal{E}$ is denoted by $\mathsf{con_f}(e_i, i, S, K)$, where $i \in \mathbb{N}$ is the ordinal of the event e_i, and $K \in \mathbb{N}$ is a parameter, called a *context size*, that limits the number of events that are considered when computing the context of the event e_i. $\square$

There are, of course, several ways of choosing the actual function $\mathsf{con_f}$. A simple solution to this problem is to define the context of an event in a sequence as a set of types of events preceding the occurrence in question. Another natural approach is to define the context of an occurrence of an event type as a sequence of events preceding it. In the following we consider both these approaches.

5.2.1 Sets of event types as contexts

We start by considering the case where the context of an occurrence of an event type is defined as a set of event types.

Definition 5.5 Let $\mathcal{E}$ be a set of event types, and $\mathcal{S}$ an event sequence over $\mathcal{E}$. A *set context* of an event (e_i, t_i) in the event sequence $\mathcal{S}$ is an unordered set of all the types of events that occur within W time units before t_i, i.e., the set

$$\mathsf{con}_{set}((e_i, t_i), \mathcal{S}, W) = \{\, e_j \mid (e_j, t_j) \in \mathcal{S} \text{ and } t_i - W \leq t_j \leq t_i \text{ and } j < i \,\}.$$

Further, a set context of an event e_i in an event type sequence S over $\mathcal{E}$ is an unordered set of all the types of the maximum K events preceding the event e_i in the sequence[1]. That is, for the event e_i, its set context with the given parameter values is the set

$$\mathsf{con}_{set}(e_i, i, S, K) = \{\, e_j \mid e_j \in S \text{ and } i - K \leq j < i \,\}.$$

An empty set context is denoted by $\emptyset$ in both cases. □

Example 5.6 Let $\mathcal{E} = \{A, B, C, D, E, F\}$ be the set of event types, and $\mathcal{S}$ the event sequence in Figure 5.1. With a window width $W = 3$, the set context of the event $(A, 38)$ in the sequence $\mathcal{S}$ is

$$\mathsf{con}_{set}((A, 38), \mathcal{S}, 3) = \{C, D\}.$$

This means that events of types C and D occurred within three time units before the event $(A, 38)$ in the sequence $\mathcal{S}$.

Then consider the event type sequence $S = \langle A, E, F, \ldots, F, E \rangle$ obtained from Figure 5.1. Now an event that corresponds to the event $(A, 38)$ in the event sequence $\mathcal{S}$ is the seventh event of the event type sequence S. Then, with a context size $K = 3$, a set context of this event is

$$\mathsf{con}_{set}(A, 7, S, 3) = \{C, D, E\}.$$

On the other hand, the context $\mathsf{con}_{set}((A, 30), \mathcal{S}, 3)$ of the event $(A, 30)$ in the sequence $\mathcal{S}$ is an empty set, as is the context $\mathsf{con}_{set}(A, 1, S, 3)$ of the corresponding event in the event type sequence S. □

Assume that we have an event sequence $\mathcal{S}$ and an event type sequence S, over the same set $\mathcal{E}$ of event types, so that they correspond to each other. As Example 5.6 shows, contexts $\mathsf{con}_{set}((e_i, t_i), \mathcal{S}, W)$ and $\mathsf{con}_{set}(e_i, i, S, K)$ obtained from such corresponding sequences $\mathcal{S}$ and S can either be the same or they can be different from each other, even if the window width W has the same numerical value as the context size K. If the original sequence is an event sequence, in most cases $\mathsf{con}_{set}((e_i, t_i), \mathcal{S}, W) \neq \mathsf{con}_{set}(e_i, i, S, K)$, although W and K have the same value. However, if the original sequence is an event type sequence, and occurrence times of the events in the corresponding event sequence are formed from the ordinals of the events, it always holds good that $\mathsf{con}_{set}((e_i, t_i), \mathcal{S}, W) = \mathsf{con}_{set}(e_i, i, S, K)$, when the values of W and K are the same. This is true regardless of the type of the event considered, the original sequence S and the parameter values W and K used.

[1]If the event e_i occurs in the beginning of the sequence S, there can be less than K events preceding it.

In the following, we use the artificial sequence of Figure 5.1 to present examples of both an event sequence and an event type sequence. In the examples from the real-life data sets, however, we study only contexts obtained from the original sequences. In other words, we consider only contexts $\mathsf{con}_{set}((e_i, t_i), \mathcal{S}, W)$ when the original sequence is an event sequence, and contexts $\mathsf{con}_{set}(e_i, i, S, K)$ when the original sequence is an event type sequence.

Example 5.7 Consider the alarm sequence $\mathcal{S}_{alarm}$ in Figure 5.2. Then a set

$$\mathsf{con}_{set}((7401, 10), \mathcal{S}_{alarm}, 5) = \{7172, 7177, 7311, 7312\}$$

is the set context of an alarm $(7401, 10)$ in the sequence $\mathcal{S}_{alarm}$ with a window width $W = 5$. □

Example 5.8 Consider the protein sequence $S_{protein}$ in Example 5.3 and the amino acid *Histidine* whose one-letter code is H. The first occurrence of H is the ninth amino acid in the sequence $S_{protein}$. With a context size $K = 5$, the set

$$\mathsf{con}_{set}(H, 9, S_{protein}, 5) = \{A, D, G, K\}.$$

represents a set context of that occurrence of the amino acid H. □

5.2.2 Sequences as contexts

Now we move on to consider another approach to defining contexts of occurrences of event types. In this approach contexts of events are defined as sequences, instead of sets of event types.

Definition 5.6 Let $\mathcal{E}$ be a set of event types, $\mathcal{S}$ an event sequence over $\mathcal{E}$, and W a window width. A *sequence context* of an event (e_i, t_i) in the event sequence $\mathcal{S}$ is defined as the event sequence preceding the occurrence of the event within the given W time units, i.e., the sequence context of the event (e_i, t_i) is the event sequence

$$\begin{aligned}\mathsf{con}_{seq}((e_i, t_i), \mathcal{S}, W) \quad = \quad & \langle\, (e_j, t_j^{'}) \mid (e_j, t_j) \in \mathcal{S},\ t_i - W \leq t_j \leq t_i, \\ & t_j^{'} = |t_i - t_j| \text{ and } j < i\,\rangle.\end{aligned}$$

A sequence context of an event e_i in an event type sequence S over $\mathcal{E}$ with a context size K, on the other hand, is defined as an event type sequence

$$\mathsf{con}_{seq}(e_i, i, S, K) = \langle\, e_j \mid e_j \in S \text{ and } i - K \leq j < i\,\rangle.$$

An empty sequence context is denoted by $\langle\rangle$ in both cases. □

Note that according to Definition 5.6, a sequence context $\mathsf{con}_{seq}(e_i, i, S, K)$ is always an event type subsequence of the sequence S [2]. This is, however, not the case with a sequence context $\mathsf{con}_{seq}((e_i, t_i), \mathcal{S}, W)$. The types of events $(e_j, t_j^{'})$ of such a sequence context are, of course, the same as in the sequence $\mathcal{S}$, but the occurrence times $t_j^{'}$ are different. Namely, each occurrence time $t_j^{'}$ of an event $(e_j, t_j^{'})$ in the sequence context is the relative time distance between the occurrence time t_j of the event (e_j, t_j) and the occurrence time t_i of the event (e_i, t_i) in the sequence $\mathcal{S}$. Therefore, the sequence context $\mathsf{con}_{seq}((e_i, t_i), \mathcal{S}, W)$ is not a real subsequence of the event sequence $\mathcal{S}$. The reason why we use the relative time distances $t_j^{'}$, instead of the times t_j, is that we want the occurrence times in all sequences contexts to be of the same magnitude. Otherwise, when comparing context sequences, sequences such as $\langle (A,5),(B,3),(C,2)\rangle$ and $\langle (A,105),(B,103),(C,102)\rangle$ would be considered to be very far from each other, even though the event types in them match exactly (see Section 4.2).

Neither the sequence context $\mathsf{con}_{seq}((e_i, t_i), \mathcal{S}, W)$ is a real event sequence; according to Definition 4.2 in Section 4.1 the events of a sequence should be in increasing temporal order. This means that if we would like the sequence context $\mathsf{con}_{seq}((e_i, t_i), \mathcal{S}, W)$ to be a real event sequence, we should reverse the order of the events in it. In the following we use, however, the context sequences as they were defined, i.e., the event of these sequences are in decreasing temporal order.

Example 5.9 Consider the event sequence $\mathcal{S}$ in Figure 5.1. The sequence context of the event $(A, 38)$ in the sequence $\mathcal{S}$ with a window width $W = 3$ is

$$\mathsf{con}_{seq}((A,38), \mathcal{S}, 3) = \langle\, (C,3), (D,1)\,\rangle.$$

On the other hand, the sequence context of the corresponding event in the event type sequence S with a context size $K = 3$ is

$$\mathsf{con}_{seq}(A, 7, S, 3) = \langle\, E, C, D\,\rangle.$$

However, sequence contexts $\mathsf{con}_{seq}((A,30), \mathcal{S}, 3)$ and $\mathsf{con}_{seq}(A, 30, S, 3)$ are both empty sequences, i.e., $\mathsf{con}_{seq}((A,30), \mathcal{S}, 3) = \mathsf{con}_{seq}(A, 30, S, 3) = \langle\rangle$. □

Example 5.10 Consider the alarm sequence $\mathcal{S}_{alarm}$ in Figure 5.2, and assume that events having the same occurrence time are written in the alarm log in such order that the event on the top comes later, i.e., that

[2] Actually $\mathsf{con}_{seq}(e_i, i, S, K)$ is an event type substring of the sequence S, i.e., it is a continuous part of the sequence S.

events $(7172, 5)$ and $(7177, 5)$ come in that order. The sequence context of the alarm $(7401, 10)$ in the sequence $\mathcal{S}_{alarm}$ is, with a window width $W = 5$, the event sequence

$$\begin{aligned}\mathsf{con}_{seq}((7401, 10), \mathcal{S}_{alarm}, 5) &= \langle\, (7172, 5), (7177, 5), (7311, 3), \\ &\quad (7312, 3), (7172, 1), (7312, 1)\,\rangle.\end{aligned}$$ □

Example 5.11 Now consider the protein sequence $S_{protein}$ in Example 5.3 and the amino acid H. As an event type sequence, the context of the first occurrence of H in the sequence $S_{protein}$ is

$$\mathsf{con}_{seq}(H, 9, S_{protein}, 5) = \langle\, A, A, D, K, G\,\rangle.$$

with the context size $K = 5$. □

5.2.3 Sets of contexts

Typically, there are several occurrences of each event type in a sequence. In computing similarity between event types we need to take into account contexts of all these occurrences. This brings us to the definition of a set of contexts.

Definition 5.7 Let $\mathcal{E}$ be a set of event types, $\mathcal{S}$ an event sequence over $\mathcal{E}$ and W a window width. Given an event type $A \in \mathcal{E}$, a *set of contexts* of all occurrences of the event type A in the sequence $\mathcal{S}$, with the window width W, is defined as

$$\mathsf{contexts}_\mathsf{f}(A, \mathcal{S}, W) = \{\, \mathsf{con}_\mathsf{f}((e_i, t_i), \mathcal{S}, W) \mid \exists\ (e_i, t_i) \in \mathcal{S} \text{ and } e_i = A\,\},$$

where f defines whether the contexts are sets of event types or sequences of events.

Then let S be an event type sequence over $\mathcal{E}$ and K a context size. Now, given an event type $A \in \mathcal{E}$, a set of contexts of the event type A in the sequence S, with the context size K, is defined as

$$\mathsf{contexts}_\mathsf{f}(A, S, K) = \{\, \mathsf{con}_\mathsf{f}(e_i, i, S, K) \mid \exists\ e_i \in S \text{ and } e_i = A\,\},$$

where f indicates the type of the contexts studied.

If there is no risk for confusion with the sequence, and the parameter value W or K, we can simply write the set of contexts of the event type A as $\mathsf{contexts}_\mathsf{f}(A)$. The size of the set of contexts is denoted by $|\mathsf{contexts}_\mathsf{f}(A)|$. Because the set the set $\mathsf{contexts}_\mathsf{f}(A)$ can contain several contexts that are exactly the same, the set $\mathsf{contexts}_\mathsf{f}(A)$ is a multiset, and its size is the same as the number of the occurrences of the event type A in the sequence. □

Example 5.12 Let $\mathcal{E} = \{A, B, C, D, E, F\}$ be the set of event types and $\mathcal{S}$ the event sequence in Figure 5.1. If contexts are defined as sets of event types and the window width is $W = 3$, the set

$$\begin{aligned}\mathsf{contexts}_{set}(A, \mathcal{S}, 3) &= \{\, \mathsf{con}_{set}((A, 30), \mathcal{S}, 3), \mathsf{con}_{set}((A, 38), \mathcal{S}, 3), \\ &\quad\ \mathsf{con}_{set}((A, 49), \mathcal{S}, 3) \,\} \\ &= \{\, \emptyset, \{C, D\}, \{D, E\} \,\},\end{aligned}$$

represents the set of contexts of the event type A in the sequence $\mathcal{S}$. Looking at the sequence with the same window width $W = 3$ and considering set contexts, we can see that the set of contexts of the event type B is exactly the same as the set of contexts of the event type A, i.e., in this case, $\mathsf{contexts}_{set}(A) = \mathsf{contexts}_{set}(B)$.

Now assume that contexts are defined as sequences. Then, with the window width $W = 3$, the set of contexts of the event type A in the event sequence $\mathcal{S}$ is the set

$$\begin{aligned}\mathsf{contexts}_{seq}(A, \mathcal{S}, 3) &= \{\, \mathsf{con}_{seq}((A, 30), \mathcal{S}, 3), \mathsf{con}_{seq}((A, 38), \mathcal{S}, 3), \\ &\quad\ \mathsf{con}_{seq}((A, 49), \mathcal{S}, 3) \,\} \\ &= \{\, \langle\rangle,\ \langle\, (C, 3), (D, 1) \,\rangle,\ \langle\, (E, 2), (D, 1) \,\rangle \,\}.\end{aligned}$$

Furthermore, the set of contexts of the event type B in the sequence $\mathcal{S}$, with the window width $W = 3$, is the set

$$\begin{aligned}\mathsf{contexts}_{seq}(B, \mathcal{S}, 3) &= \{\, \mathsf{con}_{seq}((B, 44), \mathcal{S}, 3),\ \mathsf{con}_{seq}((B, 54), \mathcal{S}, 3), \\ &\quad\ \mathsf{con}_{seq}((B, 60), \mathcal{S}, 3) \,\} \\ &= \{\langle\, (C, 2), (D, 1) \,\rangle,\ \langle\rangle,\ \langle\, (E, 2), (D, 1) \,\rangle \,\}.\end{aligned}$$

This means that now the sets of contexts of the event types A and B are nearly the same; only the occurrence times of the first events in the sequence contexts $\langle\, (C, 3), (D, 1) \,\rangle$ and $\langle\, (C, 2), (D, 1) \,\rangle$ are not equal.

Let us now study what kind of sets of contexts we obtain for the event types A and B in the corresponding event type sequence S. If contexts are defined as sets of event types, the sets of contexts of the event types A and B, with a context size $K = 3$, are

$$\begin{aligned}\mathsf{contexts}_{set}(A, S, 3) &= \{\, \mathsf{con}_{set}(A, 1, S, 3), \mathsf{con}_{set}(A, 7, S, 3), \\ &\quad\ \mathsf{con}_{set}(A, 14, S, 3) \,\} \\ &= \{\, \emptyset, \{C, D, E\}, \{B, D, E\} \,\}\end{aligned}$$

and

$$\begin{aligned}\mathsf{contexts}_{set}(B, S, 3) &= \{\, \mathsf{con}_{set}(B, 11, S, 3), \mathsf{con}_{set}(B, 16, S, 3), \\ &\quad\ \mathsf{con}_{set}(B, 21, S, 3) \,\} \\ &= \{\, \{C, D, F\}, \{A, D, F\}, \{C, D, E\} \,\}.\end{aligned}$$

In other words, the sets $\mathsf{contexts}_{set}(A, S, 3)$ and $\mathsf{contexts}_{set}(B, S, 3)$ are very different from each other. Consequently, the sets $\mathsf{contexts}_{seq}(A, S, 3)$ and $\mathsf{contexts}_{seq}(B, S, 3)$ of sequence contexts of the event types A and B cannot be similar either. □

Example 5.13 Consider the alarm sequence $\mathcal{S}_{alarm}$ in Figure 5.2. If contexts are defined as sets of event types, a set of contexts of the alarm type 7401 in the alarm sequence $\mathcal{S}_{alarm}$, with a window width $W = 5$, is the set

$$\begin{aligned}\mathsf{contexts}_{set}(7401) &= \{\, \mathsf{con}_{set}((7401, 10), \mathcal{S}_{alarm}, 5), \\ &\quad\ \mathsf{con}_{set}((7401, 22), \mathcal{S}_{alarm}, 5), \\ &\quad\ \mathsf{con}_{set}((7401, 42), \mathcal{S}_{alarm}, 5) \,\} \\ &= \{\, \{7172, 7177, 7311, 7312\},\ \{7177\},\ \{7010, 7312\} \,\}.\end{aligned}$$

On the other hand, when contexts are defined as sequences, a set

$$\begin{aligned}\mathsf{contexts}_{seq}(7401) &= \{\, \langle\, (7172, 5), (7177, 5), (7311, 3), (7312, 3), (7172, 1), \\ &\quad\ (7312, 1) \,\rangle,\ \langle\, (7177, 4) \,\rangle,\ \langle\, (7010, 4), (7312, 4) \,\rangle \,\}.\end{aligned}$$

represents the set of three sequence contexts of the alarm type 7401 with the same window width $W = 5$. □

Example 5.14 Consider the protein sequence $S_{protein}$ in Example 5.3. A set of set contexts of the amino acid H in the sequence $S_{protein}$, with a context size $K = 5$, is the set

$$\begin{aligned}\mathsf{contexts}_{set}(H) &= \{\, \mathsf{con}_{set}(H, 9, S_{protein}, 5), \\ &\quad\ \mathsf{con}_{set}(H, 20, S_{protein}, 5) \,\} \\ &= \{\, \{A, D, G, K\},\ \{G, K, V\} \,\}.\end{aligned}$$

On the other hand, a set

$$\mathsf{contexts}_{seq}(H) = \{\, \langle\, A, A, D, G, K \,\rangle, \langle\, G, K, V, G, G \,\rangle \,\}$$

of two event type sequences describes the set of sequence contexts of the amino acid H with the context size $K = 5$. □

In Definition 5.7, only contexts of occurrences of an event type in one particular sequence were taken into account. Instead of just one sequence we could, however, include in the set $\mathsf{contexts}_{\mathsf{f}}(A)$ all contexts of occurrences of the event type A in a set S of event sequences (or event type sequences). In the examples of this thesis we mainly consider contexts obtained from one sequence, but in Section 5.5 we describe experimental results on cases where the contexts are extracted from a set of sequences.

5.2.4 Variations

Above we considered only one-sided contexts of occurrences of event types. An alternative would be to define a context of an event (e_i, t_i) so that events occurring both before and after the event (e_i, t_i) were taken into account, i.e., use two-sided contexts. Then the choice to be made is, whether we consider events which occur during a time period between times $t_i - W$ and $t_i + W$, or events that occur between times $t_i - W_1$ and $t_i + W_2$, where $W_1 \neq W_2$. In the case of event type sequences a similar approach would mean considering those events that have ordinals between $i - K$ and $i + K$, or between $i - K_1$ and $i + K_2$, where $K_1 \neq K_2$. Such two-sided contexts could be useful, for example, for genome or protein data, where both directions of the sequence are equally meaningful. The main applications we consider are, however, in sequences where there is a natural direction, that of advancing time, and hence, in this thesis we concentrate on using one-sided contexts.

We would also get different contexts, if we restricted the types of events that are considered when the contexts are extracted. In extracting contexts of amino acids from protein sequences, for example, we could take into account only amino acids having certain similar biochemical properties and disregard occurrences of all other amino acids, even if they belong to the K preceding amino acids of the amino acid considered. In a way this approach resembles the choice of the probe attributes used in computing external similarities between binary attributes (see Section 3.3). In this thesis, we focus on the case where all possible types of events are taken into account when computing contexts, and hence, leave this possibility of restricting the set of event types for further study.

Yet another variation of contexts concerns the way of defining a sequence context that is obtained from an event sequence. In Definition 5.6 such a context was defined as an event sequence. We could, alternatively, define it as an event type sequence. Then, a sequence context of an event (e_i, t_i) in a sequence $\mathcal{S}$ would be an event type sequence

$$\mathsf{con}_{seq}((e_i, t_i), \mathcal{S}, W) = \langle\, e_j \mid (e_j, t_j) \in \mathcal{S},\ t_i - W \leq t_j \leq t_i, \text{ and } j < i \,\rangle.$$

But since the original definition of $\mathsf{con}_{seq}((e_i, t_i), \mathcal{S}, W)$ is in our view more natural than this new one, we cover only sequence contexts computed according to Definition 5.6 here.

5.3 Similarity between sets of contexts

In the previous section we considered some ways of defining contexts of occurrences of event types. We now move on to study how similarity between two event types could be defined as similarity between their sets of contexts. In the following, we give a general definition for such a similarity measure by using, once again, a complementary notion of distance.

Definition 5.8 Let $\mathcal{E}$ be a set of event types, and S a set of sequences over $\mathcal{E}$. If the sequences in the set S are event sequences, then the distance between two event types $A \in \mathcal{E}$ and $B \in \mathcal{E}$ is defined as

$$d_{\mathsf{F}}(A, B, \mathsf{S}) = d_{\mathsf{F}}(\mathsf{contexts_f}(A, \mathsf{S}, W), \mathsf{contexts_f}(B, \mathsf{S}, W)),$$

where W is a given window width, and $\mathsf{contexts_f}(A, \mathsf{S}, W)$ and $\mathsf{contexts_f}(B, \mathsf{S}, W)$ are the sets of contexts of the event types A and B. Further, when the sequences in the set S are event type sequences, the distance between the event types A and B, given a context size K, is defined as

$$d_{\mathsf{F}}(A, B, \mathsf{S}) = d_{\mathsf{F}}(\mathsf{contexts_f}(A, \mathsf{S}, K), \mathsf{contexts_f}(B, \mathsf{S}, K)),$$

where $\mathsf{contexts_f}(A, \mathsf{S}, K)$ and $\mathsf{contexts_f}(B, \mathsf{S}, K)$ are the sets of contexts of the event types A and B.

If there is no risk of confusing the sequence considered or the parameter value W or K used, we may in both cases write $d_{\mathsf{F}}(A, B)$. □

In Definition 5.8 we did not exactly specify what function d_{F} is used in defining the distance between two sets of contexts. In Chapters 3 and 4 we noticed that there is seldom a single notion of similarity between binary attributes or event sequences that would be natural and useful, and that the situation in hand often determines what notion to use. This conclusion also holds good with similarity between event types. Therefore, it is only natural that, when contexts of occurrences of event types are defined as sets of event types, or as sequences of events, we get different similarity notions. Furthermore, if we considered just one kind of contexts of occurrences of event types (sets or sequences), we get different similarity notions by using different functions d_{F}.

5.3.1 Distance between two sets of set contexts

We start by studying the case of set contexts. Every set context is a subset of the set $\mathcal{E}$ of all possible event types. In the following, we describe how such a context can be represented as a binary vector.

Definition 5.9 Let $\mathcal{E}$ be a set of m event types, $\mathcal{S}$ an event sequence over $\mathcal{E}$, and W a window width. A set context $\mathsf{con}_{set}((e_i,t_i),\mathcal{S},W)$ of an event $(e_i,t_i) \in \mathcal{S}$ can be represented as an m-dimensional *context vector*

$$\mathsf{v}_{\mathsf{con}}((e_i,t_i),\mathcal{S},W) = [\, u_{e_1}, u_{e_2}, \ldots, u_{e_m} \,]$$

where $e_j \in \mathcal{E}$ for all $j = 1,\ldots,m$. A coordinate u_{e_j} of the vector is 1, if the event type e_j occurs in the context $\mathsf{con}_{set}((e_i,t_i),\mathcal{S},W)$, and 0, if it does not occur in the context. The coordinates u_{e_j} are ordered in an ascending alphabetical or numerical order of the corresponding event types e_j.

Now consider an event type sequence S over $\mathcal{E}$. An m-dimensional context vector that describes a set context $\mathsf{con}_{set}(e_i,i,S,K)$ is a vector

$$\mathsf{v}_{\mathsf{con}}(e_i,i,S,K) = [\, u_{e_1}, u_{e_2}, \ldots, u_{e_m} \,] \,.$$

where $e_j \in \mathcal{E}$ for all $j = 1,\ldots,m$. In this context vector, an item u_{e_j} is also 1, if the event type e_j belongs to the set context $\mathsf{con}_{set}(e_i,i,S,K)$, and 0, if it does not.

When a set context is an empty set, it is described by an *empty context vector*, i.e., a vector $[\,0,0,\ldots,0\,]$. On the other hand, a vector $[\,1,1,\ldots,1\,]$ represents a set context that contains all the event types in the set $\mathcal{E}$. □

When all the set contexts of the occurrences of an event type are represented as context vectors, we get a collection of vectors. In the following we give a notation for such a set of context vectors.

Definition 5.10 Let $\mathcal{E}$ be a set of m event types, $\mathcal{S}$ an event sequence over $\mathcal{E}$, and W a window width. Given an event type $A \in \mathcal{E}$ and a set $\mathsf{contexts}_{set}(A,\mathcal{S},W)$, we denote by $\mathcal{V}(A,\mathcal{S},W)$ the collection of context vectors corresponding to the set $\mathsf{contexts}_{set}(A,\mathcal{S},W)$. Similarly, given an event type sequence S over $\mathcal{E}$ and a context size K, we denote by $\mathcal{V}(A,S,K)$ the collection of context vectors describing set contexts of a set $\mathsf{contexts}_{set}(A,S,K)$. If the sequence and the parameter values W and K are unmistakable, we may in both cases use the abbreviation $\mathcal{V}(A)$. Such a vector set is a multiset, and its size, that is denoted by $|\mathcal{V}(A)|$, is the same as the number of contexts in the set $\mathsf{contexts}_{set}(A)$. □

Example 5.15 Let $\mathcal{E} = \{A,B,C,D,E,F\}$ be the set of event types, and $\mathcal{S}$ the event sequence in Figure 5.1. The set $\mathsf{contexts}_{set}(A,\mathcal{S},3)$ given in Example 5.12 would now be presented as a set of context vectors as

$$\begin{aligned} \mathcal{V}(A) &= \{\, \mathsf{v}_{\mathsf{con}}((A,30),\mathcal{S},3),\ \mathsf{v}_{\mathsf{con}}((A,38),\mathcal{S},3), \\ &\qquad \mathsf{v}_{\mathsf{con}}((A,49),\mathcal{S},3) \,\} \\ &= \{\, [0,0,0,0,0,0]\,,\ [0,0,1,1,0,0]\,,\ [0,0,0,1,1,0] \,\}. \end{aligned}$$

Exactly the same set of vectors would be obtained, if we had considered the set $\mathsf{contexts}_{set}(B, \mathcal{S}, 3)$ of contexts of the event type B.

Consider then the corresponding event type sequence S and a context size $K = 3$. Now the sets

$$\mathcal{V}(A) = \{\, [0,0,0,0,0,0]\,,\ [0,0,1,1,1,0]\,,\ [0,1,0,1,1,0]\,\}$$

and

$$\mathcal{V}(B) = \{\, [0,0,1,1,0,1]\,,\ [1,0,0,1,0,1]\,,\ [0,0,1,1,1,0]\,\}$$

of context vectors would represent the sets $\mathsf{contexts}_{set}(A, S, 3)$ and $\mathsf{contexts}_{set}(B, S, 3)$ given in Example 5.12, respectively. □

Assume that set contexts of occurrences of event types are represented as m-dimensional vectors. Then a similarity between two such vectors could, for example, be defined as the number of positions in which the vectors differ, i.e., as the *Hamming distance* [Ham50] of the vectors. This measure corresponds to using a symmetric difference between the sets of event types. Also any of the Minkowski metrics [Nii87, KR90] could be used as a distance measure between two context vectors. Given two event types A and $B \in \mathcal{E}$, their sets $\mathcal{V}(A)$ and $\mathcal{V}(B)$ of context vectors are, however, two sets of vectors in an m-dimensional space. Therefore, we have to define similarity between these sets, not just between two context vectors in them.

A statistical approach to defining a distance between two sets of context vectors would be to view the sets $\mathcal{V}(A)$ and $\mathcal{V}(B)$ as samples from two distributions g_A and g_B of the m-dimensional hypercube and define similarity between the event types A and B as similarity between g_A and g_B. This can, in turn, be defined for example by using the Kullbach-Leibler distance [KL51, Kul59, Bas89]:

$$d(g_A \,\|\, g_B) = \sum_{x \in \{0,1\}^m} g_A(x) \cdot \log \frac{g_B(x)}{g_A(x)}$$

or its symmetrized version $d(g_A \,\|\, g_B) + d(g_B \,\|\, g_A)$. In these measures the summation variable x varies over all of the 2^m points of the hypercube, and hence, direct application of the formula is not feasible.

Another related alternative is to view the set $\mathcal{V}(A)$ as a sample from a multivariate normal distribution and compute the likelihood of obtaining the set $\mathcal{V}(B)$ as a sample from the same distribution. For determining such a likelihood, we need the following.

Definition 5.11 Let $\mathcal{E}$ be a set of m event types, $\mathcal{S}$ an event sequence over $\mathcal{E}$, and W a window width. Then assume that we have an event type

$A \in \mathcal{E}$, and a set $\mathcal{V}(A, \mathcal{S}, W)$ of context vectors of the event type A. For each $e_j \in \mathcal{E}$, the mean of the values u_{e_j} in the vectors of the set $\mathcal{V}(A, \mathcal{S}, W)$ is denoted by

$$\mu_{e_j}^A = \frac{\sum_{i=1}^k u_{i,e_j}}{k},$$

and the variance of the values u_{e_j} by

$$\sigma_{e_j}^A = \frac{\sum_{i=1}^k (u_{i,e_j} - \mu_{e_j}^A)^2}{k},$$

when $k = |\mathcal{V}(A, \mathcal{S}, W)|$.

Now consider an event type sequence S over $\mathcal{E}$ and a context size K. Then, in the same way as above, we denote by $\mu_{e_j}^A$ the mean of the values u_{e_j} in the vectors of the set $\mathcal{V}(A, S, K)$, and by $\sigma_{e_j}^A$ the variance of the values u_{e_j} in the vectors of the set $\mathcal{V}(A, S, K)$. □

Assume that we have a set $\mathcal{V}(A)$ and, for every event type $C \in \mathcal{E}$, the values μ_C^A and σ_C^A. Given a vector $\mathsf{v}_{\mathsf{con},i} \in \mathcal{V}(B)$, the likelihood of obtaining the vector $\mathsf{v}_{\mathsf{con},i}$ from the distribution of the set $\mathcal{V}(A)$ is proportional to

$$\prod_{C \in \mathcal{E}} \exp\left(-((u_{i,C} - \mu_C^A)^2 / \sigma_C^A)\right) = \exp\left(-\sum_{C \in \mathcal{E}} ((u_{i,C} - \mu_C^A)^2 / \sigma_C^A)\right).$$

Using this, the logarithmic likelihood $\ell_{\log}(B|A)$ of the whole set $\mathcal{V}(B)$ is

$$\ell_{\log}(B|A) = -\sum_{i=1}^{k} \sum_{C \in \mathcal{E}} ((u_{i,C} - \mu_C^A)^2 / \sigma_C^A),$$

where $k = |\mathcal{V}(B)|$. This formula or its symmetrized version, $\ell_{\log}(B|A) + \ell_{\log}(A|B)$, could now be used as a distance measure between event types.

A problem with this approach of likelihood functions is that these measures can impose a high value of dissimilarity on the basis of a single event type. Namely, if we have for some event type $C \in \mathcal{E}$ such that the set $\mathsf{contexts}_{set}(A)$ contains no set with the event type C in it, then the values of both the mean μ_C^A and the variance σ_C^A are zero. If now at least one context of the event type B contains C, we have $\ell_{\log}(B|A) = -\infty$, indicating that the event type B is very far from the event type A. In a way this conclusion is, of course, justified: no context of the event type A included the event type C, but at least one context of the event type B did. However, in most cases we would not like to draw such an extreme conclusion on the basis of difference in one event type.

A way of alleviating this problem is to use some assumptions of the presence of an event type in a context. This could, for example, be done by adding to each set of contexts an empty context and a context containing all the event types in $\mathcal{E}$, i.e., by adding vectors $[0, 0, \ldots, 0]$ and $[1, 1, \ldots, 1]$ to each set $\mathcal{V}$. Then the variance σ_C^A cannot be zero for any event type $C \in \mathcal{E}$, and the value of the likelihood function $\ell_{\log}$ cannot be infinite only because of one event type.

Our solution for defining similarity between event types when their contexts are sets of event types is, however, even simpler: we first compute for each set of contexts a centroid vector, and then define a distance between two event types as a distance between the centroid vectors of their sets of contexts. We first define the centroid vector of a set of contexts.

Definition 5.12 Let $\mathcal{E}$ be a set of m event types, and A an event type in the set $\mathcal{E}$. Given an event sequence $\mathcal{S}$ over $\mathcal{E}$, a window width W, and values $\mu_{e_j}^A$, for all $e_j \in \mathcal{E}$, we can identify each set $\mathcal{V}(A, \mathcal{S}, W)$ with its *centroid vector* $\mathsf{cev}(A, \mathcal{S}, W)$, i.e., a vector

$$\mathsf{cev}(A, \mathcal{S}, W) = \left[\mu_{e_1}^A, \mu_{e_2}^A, \ldots, \mu_{e_m}^A\right].$$

Similarly, given an event type sequence S over $\mathcal{E}$, a context size K, and values $\mu_{e_j}^A$, for all $e_j \in \mathcal{E}$, a centroid vector of a set $\mathcal{V}(A, S, K)$ is a vector $\mathsf{cev}(A, S, K) = \left[\mu_{e_1}^A, \mu_{e_2}^A, \ldots, \mu_{e_m}^A\right]$. If the sequence and the parameter values W and K considered are obvious, we use an abbreviation $\mathsf{cev}(A)$ for both types of centroid vectors. □

Example 5.16 Consider the set $\mathcal{E} = \{A, B, C, D, E, F\}$ of event types, and the sets $\mathcal{V}$ of context vectors in Example 5.15. The centroid vector of the set $\mathcal{V}(A, \mathcal{S}, 3)$ is

$$\mathsf{cev}(A, \mathcal{S}, 3) = \left[0, 0, \frac{1}{3}, \frac{2}{3}, \frac{1}{3}, 0\right],$$

which is the same as the centroid vector $\mathsf{cev}(B, \mathcal{S}, 3)$. On the other hand, a centroid vector of the set $\mathcal{V}(A, S, 3)$ is a vector $\mathsf{cev}(A, S, 3) = \left[0, \frac{1}{3}, \frac{1}{3}, \frac{2}{3}, \frac{2}{3}, 0\right]$, whereas a vector $\mathsf{cev}(A, S, 3) = \left[\frac{1}{3}, 0, \frac{2}{3}, 1, \frac{1}{3}, \frac{2}{3}\right]$ represents the centroid vector of the set $\mathcal{V}(B, S, 3)$. □

Now we are ready to define similarity between two event types when their contexts are sets of event types. We define their similarity as a distance between the centroid vectors of their sets of contexts.

Definition 5.13 Let $\mathcal{E}$ be a set of event types, A and B two event types in the set $\mathcal{E}$, and $\mathsf{contexts}_{set}(A)$ and $\mathsf{contexts}_{set}(B)$ their sets of set contexts in an event sequence $\mathcal{S}$ over $\mathcal{E}$, with a given window width W, or in an event type sequence S over $\mathcal{E}$, with a context size K. When the sets $\mathsf{contexts}_{set}(A)$ and $\mathsf{contexts}_{set}(B)$ are described by centroid vectors $\mathsf{cev}(A)$ and $\mathsf{cev}(B)$, a *centroid vector distance* between the event types A and B is defined as

$$d_{\mathsf{cev}}(A,B) = |\,\mathsf{cev}(A) - \mathsf{cev}(B)\,| = \sum_{C \in \mathcal{E}} |\,\mu_C^A - \mu_C^B\,|.$$

□

The domain of the distance measure d_{cev} is $[0, m]$, where m is the number of event types in the set $\mathcal{E}$ considered. This distance measure resembles the Manhattan distance [Nii87, KR90], that is known to be a metric. The measure d_{cev} is, however, only a pseudometric, because its value can be zero even if the event types compared are not identical, i.e., $A \neq B$. This happens when the centroid vectors $\mathsf{cev}(A)$ and $\mathsf{cev}(B)$ are the same. Note that, even if the centroid vectors $\mathsf{cev}(A)$ and $\mathsf{cev}(B)$ are identical, the sets $\mathsf{contexts}_{set}(A)$ and $\mathsf{contexts}_{set}(B)$ can still differ from each other.

Example 5.17 Consider the event type set $\mathcal{E} = \{A, B, C, D, E, F\}$, and the event sequence $\mathcal{S}$ in Figure 5.1. As we have seen in Example 5.16, the sets of contexts of the event types A and B in the sequence $\mathcal{S}$ with a window width $W = 3$ are exactly the same. Therefore, also the centroid vectors of their sets of contexts are the same, and the centroid vector distance between these two event types is $d_{\mathsf{cev}}(A,B) = 0$. However, for the corresponding event type sequence S and a context size $K = 3$, the centroid vector distance between the event types A and B is

$$\begin{aligned} d_{\mathsf{cev}}(A,B) &= |\,\mathsf{cev}(A) - \mathsf{cev}(B)\,| \\ &= |\left[0, \tfrac{1}{3}, \tfrac{1}{3}, \tfrac{2}{3}, \tfrac{2}{3}, 0\right] - \left[\tfrac{1}{3}, 0, \tfrac{2}{3}, 1, \tfrac{1}{3}, \tfrac{2}{3}\right]| \\ &= \tfrac{1}{3} + \tfrac{1}{3} + \tfrac{1}{3} + \tfrac{1}{3} + \tfrac{1}{3} + \tfrac{2}{3} = 2\tfrac{1}{3}. \end{aligned}$$

This indicates that the sets $\mathsf{contexts}_{set}(A, S, 3)$ and $\mathsf{contexts}_{set}(B, S, 3)$ are not particularly similar.

Then consider the event type $E \in \mathcal{E}$. The set of contexts of the event type E in the sequence $\mathcal{S}$, with the window width $W = 3$, is

$$\begin{aligned} \mathsf{contexts}_{set}(E, \mathcal{S}, 3) &= \{\,\mathsf{con}_{set}((E,33), \mathcal{S}, 3),\ \mathsf{con}_{set}((E,47), \mathcal{S}, 3), \\ &\qquad \mathsf{con}_{set}((E,58), \mathcal{S}, 3),\ \mathsf{con}_{set}((E,66), \mathcal{S}, 3)\,\} \\ &= \{\,\{A, D, F\},\ \{B\},\ \{C, F\},\ \{D, F\}\,\}. \end{aligned}$$

For this set of contexts the centroid vector is $\mathsf{cev}(E) = \left[\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{2}, 0, \frac{3}{4}\right]$. The centroid vector distance between the event types A and E (and between the event types B and E) is then

$$d_{\mathsf{cev}}(A,E) = \mid \left[0, 0, \frac{1}{3}, \frac{2}{3}, \frac{1}{3}, 0\right] - \left[\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{2}, 0, \frac{3}{4}\right] \mid = 1\frac{5}{6}.$$

On the other hand, if we look at the corresponding event type sequence S and use the context size $K = 3$, the set of contexts of the event type E is

$$\begin{aligned} \mathsf{contexts}_{set}(E,S,3) &= \{\, \mathsf{con}_{set}(E,4,S,3),\ \mathsf{con}_{set}(E,12,S,3), \\ &\qquad \mathsf{con}_{set}(E,19,S,3),\ \mathsf{con}_{set}(E,25,S,3)\,\} \\ &= \{\, \{A,D,F\},\ \{B,C,D\},\ \{B,C,F\},\ \{C,D,F\}\,\}. \end{aligned}$$

For this set of contexts we get a centroid vector $\mathsf{cev}(E) = \left[\frac{1}{4}, \frac{1}{2}, \frac{3}{4}, \frac{3}{4}, 0, \frac{3}{4}\right]$. Now the centroid vector distance between the event types A and E is

$$d_{\mathsf{cev}}(A,E) = \mid \left[0, \frac{1}{3}, \frac{1}{3}, \frac{2}{3}, \frac{2}{3}, 0\right] - \left[\frac{1}{4}, \frac{1}{2}, \frac{3}{4}, \frac{3}{4}, 0, \frac{3}{4}\right] \mid = 2\frac{1}{3},$$

whereas the distance between the event types B and E is

$$d_{\mathsf{cev}}(B,E) = \mid \left[\frac{1}{3}, 0, \frac{2}{3}, 1, \frac{1}{3}, \frac{2}{3}\right] - \left[\frac{1}{4}, \frac{1}{2}, \frac{3}{4}, \frac{3}{4}, 0, \frac{3}{4}\right] \mid = 1\frac{1}{3}.$$

If we compare these centroid vector distances between the chosen event types, we can see that when the set contexts are extracted from the event sequence $\mathcal{S}$, the event types A and B are more similar than the other two pairs of event types. On the other hand, when the set contexts are extracted from the event type sequence S, the most similar pair of event types are the event types B and E. From this we can conclude that similarity between two event types may depend very much on, whether their contexts are extracted from an event sequence, or an event type sequence. □

The centroid vector distance d_{cev} has the advantage of being a robust measure in the sense that a single event type cannot have an unbounded effect on its value. This measure is also very fast to compute. Its drawback is that the sizes of the sets $\mathsf{contexts}_{set}(A)$ and $\mathsf{contexts}_{set}(B)$ are not directly taken into account in computing it. However, if the sets of contexts considered are of the same magnitude, this problem is not severe.

Instead of the Manhattan distance of centroid vectors we could use any other measure from the family of Minkowski metrics [Nii87, KR90] to define the distance between centroid vectors. Other possible measures considering a distance between two vectors could also be used as a distance measure between two centroid vectors. Analyzing effects of these variations of the centroid vector distance is, however, left for future study.

5.3.2 Distance between two sets of sequence contexts

We now move on to study how similarity, or distance between two event types could be defined when their contexts are defined as sequences, instead of unordered sets of event types. To make the rest of the definitions in this section more legible, we start by giving a simpler notation for sequence contexts.

Definition 5.14 Let $\mathcal{E}$ be the set of event types, $\mathcal{S}$ an event sequence over $\mathcal{E}$, and W a window width. Assume that we have an event type $A \in \mathcal{E}$ and a set $\mathsf{contexts}_{seq}(A, \mathcal{S}, W)$ of sequence contexts of the event type A with the window width W. Assuming also that the sequence contexts in the set $\mathsf{contexts}_{seq}(A, \mathcal{S}, W)$ have some kind of an order, a sequence context $\mathsf{con}_{seq}((e_i, t_i), \mathcal{S}, W)$ in the set $\mathsf{contexts}_{seq}(A, \mathcal{S}, W)$ is denoted by $\mathcal{S}_a$, where $a \in \{1, 2, \ldots, |\mathsf{contexts}_{seq}(A, \mathcal{S}, W)|\}$ is the ordinal of this sequence context in the given order.

Now let S be an event type sequence over $\mathcal{E}$, A an event type in the set $\mathcal{E}$, and K a context size. Given a set $\mathsf{contexts}_{seq}(A, S, K)$ of sequence contexts of the event type A, with the context size K, in some kind of an order, a sequence context $\mathsf{con}_{seq}(e_i, i, S, K)$ in the set $\mathsf{contexts}_{seq}(A, S, K)$ is denoted by S_a, where $a \in \{1, 2, \ldots, |\mathsf{contexts}_{seq}(A, S, K)|\}$ is the ordinal of this sequence context in the given order. $\square$

Intuitively, it is clear that a measure determining the distance between sets of sequence contexts should be based on similarities or distances between sequences in those sets. A distance between two event sequences, or similarly between event type sequences, can be computed using an edit distance type of an approach, for example, as studied in Chapter 4. Determining a distance between sets of sequences is, however, more complicated.

One possibility to solve this problem is, of course, to consider an approach resembling the centroid vector distance used in the case of set contexts. Representing event sequences and event type sequences as m-dimensional vectors would be quite difficult and quite unnatural. Instead of that, we could try to find some kind of a *centroid sequence* for each set of sequence contexts, but finding such a sequence may be difficult. Assume that a centroid sequence of the set $\mathsf{contexts}_{seq}(A)$ of sequence contexts of an event type A is defined as a maximal common subsequence of all the sequence contexts in the set $\mathsf{contexts}_{seq}(A)$, i.e., as a sequence of those events that are common to all the sequence contexts in the set $\mathsf{contexts}_{seq}(A)$. If now the sequence contexts in the set $\mathsf{contexts}_{seq}(A)$ are very different from each other, the resulting centroid sequence may easily be an empty sequence. And when two sets of contexts are both identified with an empty centroid sequence, the distance between these centroid sequences is zero, even though their sets of contexts in fact can be completely different. Moreover, if there are many such pairs of event types, we cannot get any real information about distances or similarities between the event types. Alas, the measure would be worthless.

A more appropriate approach for defining similarity between sets of sequence contexts is based on computing pairwise distances between the sequences in two sets of contexts, and then determining the distance between two sets of sequence contexts as some function of these pairwise distances. This leads us to the following definition.

Definition 5.15 Let $\mathcal{E}$ be a set of event types, A and B two event types in the set $\mathcal{E}$, and $\mathsf{contexts}_{seq}(A)$ and $\mathsf{contexts}_{seq}(B)$ their sets of sequence contexts. If the sequence contexts are event sequences, then the distance between the event types A and B is defined as

$$d_{\mathsf{F}}(A,B) = \mathsf{F}(\{\, d(\mathcal{S}_a, \mathcal{S}_b) \mid \mathcal{S}_a \in \mathsf{contexts}_{seq}(A) \text{ and } \mathcal{S}_b \in \mathsf{contexts}_{seq}(B) \,\}),$$

where F is a function of the set of pairwise distances between the event sequences in the sets $\mathsf{contexts}_{seq}(A)$ and $\mathsf{contexts}_{seq}(B)$. Similarly, when the sequence contexts in the sets $\mathsf{contexts}_{seq}(A)$ and $\mathsf{contexts}_{seq}(B)$ are event type sequences, the distance between the event types A and B is defined as

$$d_{\mathsf{F}}(A,B) = \mathsf{F}(\{\, d(S_a, S_b) \mid S_a \in \mathsf{contexts}_{seq}(A) \text{ and } S_b \in \mathsf{contexts}_{seq}(B) \,\}).$$

□

The function F in Definition 5.15 is used for combining the pairwise distances between sequences belonging to two different sets of sequence contexts. The choice of the function F is, however, not obvious. Three simple alternatives would be functions, by using which, the distance between two event types would be defined as the *minimum*, the *maximum*, or the *average* of the pairwise distances between the sequences in their sets of contexts. In the following, we denote the corresponding distance measures of event type similarity by $d_{\min}$, $d_{\max}$ and d_{avg}, respectively.

Now we give some examples of distances $d_{\min}$, $d_{\max}$ and d_{avg} between sets of context sequences. In these examples, as in all the other examples after them, the pairwise distances between context sequences are their edit distances. In computing these edit distances we used the unit operation costs, especially in the case where the sequence contexts are event sequences, the unit operation costs with the parameter value $V = \frac{1}{W}$, where W is the window width used in extracting the sequence contexts. All the edit distances were also normalized in the way explained in Chapter 4.

Example 5.18 Consider the event type set $\mathcal{E} = \{A, B, C, D, E, F\}$, and the event sequence $\mathcal{S}$ in Figure 5.1. Using the sets $\mathsf{contexts}_{seq}(A)$ and $\mathsf{contexts}_{seq}(B)$ of event sequences given in Example 5.12, the distance $d_{\min}$ between the event types A and B is

$$d_{\min}(A, B) = \min\{1, 0, 1, 0.08, 1, 0.50, 0.50, 1, 0\} = 0.$$

This means that according to this measure $d_{\min}$ the event types A and B are exactly alike. This result does not seem very natural, because when we look at the corresponding sets of sequence contexts, they really are very similar, but not exactly the same. The distance d_{avg} between the sets $\mathsf{contexts}_{seq}(A)$ and $\mathsf{contexts}_{seq}(B)$ would, however, be 0.56, meaning that the event types A and B are considered neither very similar nor very dissimilar. This result is natural in a way, because the sequences within each of the sets of contexts are not particularly homogeneous. On the other hand, the result $d_{\max}(A, B) = 1$, which means that the event types A and B are completely dissimilar, does not seem to be very reasonable at all.

Then consider the corresponding event type sequence S and the sets

$$\begin{aligned} \mathsf{contexts}_{seq}(A, S, 3) &= \{\, \mathsf{con}_{seq}(A, 1, S, 3), \mathsf{con}_{seq}(A, 7, S, 3), \\ &\quad\ \mathsf{con}_{seq}(A, 14, S, 3)\,\} \\ &= \{\, \langle\rangle, \langle E, C, D\rangle, \langle B, E, D\rangle\,\} \end{aligned}$$

and

$$\begin{aligned} \mathsf{contexts}_{seq}(B, S, 3) &= \{\, \mathsf{con}_{seq}(B, 11, S, 3), \mathsf{con}_{seq}(B, 16, S, 3), \\ &\quad\ \mathsf{con}_{seq}(B, 21, S, 3)\,\} \\ &= \{\, \langle F, C, D\rangle, \langle D, A, F\rangle, \langle C, E, D\rangle\,\}. \end{aligned}$$

that are obtained from the sequence S using the context size $K = 3$. With these sets of event type sequences, the event types A and B have distances $d_{\min}(A, B) = 0.33$, $d_{\max}(A, B) = 1$, and $d_{\text{avg}}(A, B) = 0.67$. This means that according to the measure $d_{\min}$ the event types are more similar than dissimilar, whereas they are more dissimilar than similar according to the measure d_{avg}. On the other hand, according to the measure $d_{\max}$ these attributes are completely dissimilar. When we look at the sequences in the sets $\mathsf{contexts}_{seq}(A)$ and $\mathsf{contexts}_{seq}(B)$, we can notice that they are rather different, although they have some similarities. Thus, it seems that, in this case, the measure d_{avg} of these three measures describes the similarity between the event types A and B in the best way. □

The distance measures $d_{\min}$, $d_{\max}$, and d_{avg} are simple, but using them is not really an alternative when we want to define similarity between event types. Namely, there are rather obvious problems with each of them. First consider the measure $d_{\min}$. If we now want to define similarity between two event types A and B and there is at least one empty sequence in both sets $\mathsf{contexts}_{seq}(A)$ and $\mathsf{contexts}_{seq}(B)$, the distance $d_{\min}$ between these event types is zero. In real-life data sets this can be true for many pairs of event types, and therefore, the measure $d_{\min}$ cannot find any differences between the pairs of event types.

On the other hand, if we consider a sequence in the set $\mathsf{contexts}_{seq}(A)$ and compare it to the sequences in the set $\mathsf{contexts}_{seq}(B)$, it is almost certain that there is at least one sequence in the set $\mathsf{contexts}_{seq}(B)$ so that the edit distance between these two sequences is one, i.e., these sequences have nothing in common. In such a case, the distance measure $d_{\max}$ says that these two event types are completely dissimilar, i.e., $d_{\max}(A, B) = 1$, and in the set of event types, there can easily be many such pairs of event types. This, in turn, means that neither the measure $d_{\max}$ can find any differences between the pairs of event types. Because such a result is by no means desirable, the measure $d_{\max}$ is not a good choice for the measure of the distance between event types.

At first glance, the distance measure d_{avg} seems to be a better alternative for such a distance measure. However, if the sets of contexts of two event types A and B are equal, the distance $d_{\text{avg}}(A, B)$ is not zero, as one would expect. The value of the measure d_{avg} can be zero only when both the sets $\mathsf{contexts}_{seq}(A)$ and $\mathsf{contexts}_{seq}(B)$ contain only one equal sequence, or when there are several sequences in these sets and all these sequences are identical. In real-life data sets, such situations, however, occur very seldom, and thus, neither the measure d_{avg} is a particularly good similarity measure for our purposes.

Instead of the simple functions F above, we could also consider more complex ways of defining the distance between sets of sequence contexts. One such an approach is based on using a concept of the minimum distance between a sequence in one of the sets of contexts, and the other set of contexts.

Definition 5.16 Let $\mathcal{E}$ be a set of event types, A and B two event types in the set $\mathcal{E}$, and $\mathsf{contexts}_{seq}(A)$ and $\mathsf{contexts}_{seq}(B)$ the sets of contexts of the event types A and B. The minimum distance between an event sequence $\mathcal{S}_a \in \mathsf{contexts}_{seq}(A)$ and the set $\mathsf{contexts}_{seq}(B)$ is defined as

$$d_{md}(\mathcal{S}_a, \mathsf{contexts}_{seq}(B)) = \min \{ \, d(\mathcal{S}_a, \mathcal{S}_b) \mid \mathcal{S}_b \in \mathsf{contexts}_{seq}(B) \, \}.$$

Similarly, when sequence contexts are event type sequences,

$$d_{md}(S_a, \mathsf{contexts}_{seq}(B)) = \min \{ \, d(S_a, S_b) \mid S_b \in \mathsf{contexts}_{seq}(B) \, \}$$

is the distance between an event type sequence $S_a \in \mathsf{contexts}_{seq}(A)$ and the set $\mathsf{contexts}_{seq}(B)$. If there is no risk for confusion, we may use abbreviations $d_{md}(\mathcal{S}_a, B)$ and $d_{md}(S_a, B)$ for these minimum distances. □

Using the minimum distances $d_{md}(\mathcal{S}_a, B)$ for each $\mathcal{S}_a \in \mathsf{contexts}_{seq}(A)$, and similarly, the minimum distances $d_{md}(\mathcal{S}_b, A)$ for each $\mathcal{S}_b \in \mathsf{contexts}_{seq}(B)$, we can define the distance between the sets $\mathsf{contexts}_{seq}(A)$ and $\mathsf{contexts}_{seq}(B)$ as some function of these minimum distances. One well-known measure that uses such minimum distances is the *Hausdorff distance* [Nii87, EM97]. In the case of event sequence contexts, this distance between the event types A and B is

$$d_h(A, B) = \max\{\max_{\mathcal{S}_a}\{d_{md}(\mathcal{S}_a, B)\},\ \max_{\mathcal{S}_b}\{d_{md}(\mathcal{S}_b, A)\}\},$$

where each $\mathcal{S}_a$ is an event sequence in the set $\mathsf{contexts}_{seq}(A)$ and $\mathcal{S}_b$ is an event sequence in the set $\mathsf{contexts}_{seq}(B)$. Similarly, in the case of event type sequence contexts, the Hausdorff distance between the event types A and B is

$$d_h(A, B) = \max\{\max_{S_a}\{d_{md}(S_a, B)\},\ \max_{S_b}\{d_{md}(S_b, A)\}\}$$

where each S_a is an event type sequence in the set $\mathsf{contexts}_{seq}(A)$ and S_b is an event type sequence in the set $\mathsf{contexts}_{seq}(B)$.

Example 5.19 Consider the event type set $\mathcal{E} = \{A, B, C, D, E, F\}$, and the event sequence $\mathcal{S}$ in Figure 5.1. Using the sets $\mathsf{contexts}_{seq}(A)$ and $\mathsf{contexts}_{seq}(B)$ of event sequence contexts given in Example 5.12, the Hausdorff distance d_h between the event types A and B is

$$\begin{aligned} d_h(A, B) &= \max\{\max\{0, 0.083, 0\},\ \max\{0.083, 0, 0\}\} \\ &= \max\{0.083, 0.083\} = 0.083. \end{aligned}$$

According to this measure the event types A and B are rather similar, which is a natural result. However, if we use the sets of sequence contexts obtained from the corresponding event type sequence S and given in Example 5.18, the Hausdorff distance between the event types A and B is

$$\begin{aligned} d_h(A, B) &= \max\{\max\{1, 0.333, 0.333\},\ \max\{0.333, 0.667, 0.333\}\} \\ &= \max\{1, 0.667\} = 1, \end{aligned}$$

which means that according to this measure, the event types A and B are said to be completely dissimilar. □

The Hausdorff measure is known to be a metric [Nii87], and it is used in many application areas such as computational geometry (e.g., [HK90, HKK92]) and image matching (e.g., [HR92]). However, because it is very sensitive to outliers, as the second part of Example 5.19 shows, it is not a good choice for a distance measure between sets of sequence contexts. To alleviate the problem with outliers, we use the following measure as the distance between the sets of sequence contexts.

Definition 5.17 Let $\mathcal{E}$ be the event type set, A and B two event types in the set $\mathcal{E}$, and $\mathsf{contexts}_{seq}(A)$ and $\mathsf{contexts}_{seq}(B)$ their sets of sequence contexts. If the contexts are event sequences, and we have all the minimum distances $d_{md}(\mathcal{S}_a, B)$ and $d_{md}(\mathcal{S}_b, A)$, for each $\mathcal{S}_a \in \mathsf{contexts}_{seq}(A)$ and each $\mathcal{S}_b \in \mathsf{contexts}_{seq}(B)$, then an *average minimum distance* between the event types A and B is defined as

$$d_{\mathsf{avm}}(A, B) = \frac{1}{2} \cdot (\operatorname*{avg}_{\mathcal{S}_a} \{d_{md}(\mathcal{S}_a, B)\} + \operatorname*{avg}_{\mathcal{S}_b} \{d_{md}(\mathcal{S}_b, A)\}).$$

Similarly, when the contexts are event type sequences, we denote the distance between the event types A and B by

$$d_{\mathsf{avm}}(A, B) = \frac{1}{2} \cdot (\operatorname*{avg}_{S_a} \{d_{md}(S_a, B)\} + \operatorname*{avg}_{S_b} \{d_{md}(S_b, A)\}).$$

□

To obtain the measure d_{avm}, the means of the minimum distances $d_{md}(\mathcal{S}_a, B)$ and $d_{md}(\mathcal{S}_b, A)$ are added, and this sum is normalized to give distance values between zero and one. The extreme value zero is obtained only when the sets $\mathsf{contexts}_{seq}(A)$ and $\mathsf{contexts}_{seq}(B)$ are identical, and the other extreme value one, when the sequence contexts in the two sets have nothing in common. The measure d_{avm} is also a symmetric measure. Unfortunately, it does not fulfill the triangle inequality property, and thus, it is only a semimetric. As already stated in Chapter 2, it still can be used as a distance measure between sets of sequence contexts of event types.

Example 5.20 Consider the event type set $\mathcal{E} = \{A, B, C, D, E, F\}$, and the event sequence $\mathcal{S}$ in Figure 5.1. The sets of the contexts of the event types A and $B \in \mathcal{E}$ as event sequences were given in Example 5.12. With the given sets $\mathsf{contexts}_{seq}(A)$ and $\mathsf{contexts}_{seq}(B)$, the distance d_{avm} between the event types A and B is

$$\begin{aligned} d_{\mathsf{avm}}(A, B) &= \tfrac{1}{2} \cdot (\operatorname*{avg}_{\mathcal{S}_a} \{d_{md}(\mathcal{S}_a, B)\} + \operatorname*{avg}_{\mathcal{S}_b} \{d_{md}(\mathcal{S}_b, A)\}) \\ &= \tfrac{1}{2} \cdot (\operatorname{avg} \{0, 0.083, 0\} + \operatorname{avg} \{0.083, 0, 0\}) \\ &= \tfrac{1}{2} \cdot (0.028 + 0.028) = 0.028. \end{aligned}$$

This means that the event types A and B are very similar according to this measure. This is a natural result, as the sets of contexts are almost the same.

If we then look at the sets of the sequence contexts of the event types A and B obtained from the corresponding event type sequence S and given in Example 5.18, the average minimum distance between the event types A and B is

$$\begin{aligned} d_{\mathsf{avm}}(A,B) &= \tfrac{1}{2} \cdot (\operatorname{avg}\{1, 0.333, 0.333\} + \operatorname{avg}\{0.333, 0.667, 0.333\}) \\ &= \tfrac{1}{2} \cdot (0.556 + 0.444) = 0.500. \end{aligned}$$

Now, according to this measure, the event types A and B are considered to be neither similar nor very dissimilar, which well describes the similarity between their sets of contexts. □

5.4 Algorithms for computing event type similarity

In this section we present algorithms needed for computing similarities between event types. We also briefly study the time and space complexities of these algorithms.

Algorithm for extracting contexts of events

We start by presenting Algorithm 5.1, which is used to extract contexts of occurrences of event types from an event sequence.

Algorithm 5.1 Contexts of event types

Input: A set $\mathcal{E}$ of event types, a set $\mathcal{EI}$ of interesting event types, an event sequence $\mathcal{S}$ over $\mathcal{E}$ and a window width W.

Output: Sets of contexts $\mathsf{contexts_f}(A, \mathcal{S}, W)$ for all $A \in \mathcal{EI}$.

Method:

1. **for** all event types $A \in \mathcal{EI}$ **do**
2.   **for** all events e_i of type A in $\mathcal{S}$ **do**
3.   compute $\mathsf{con_f}((e_i, t_i), \mathcal{S}, W)$;
4.   add $\mathsf{con_f}((e_i, t_i), \mathcal{S}, W)$ in the set $\mathsf{contexts_f}(A, \mathcal{S}, W)$;
5.   **od**;
6. **od**;
7. output the sets $\mathsf{contexts_f}(A, \mathcal{S}, W)$ of contexts for each A in $\mathcal{EI}$;

The input of Algorithm 5.1 are a set $\mathcal{E}$ of all possible event types, and a set $\mathcal{EI}$ of interesting event types when $\mathcal{EI} \subseteq \mathcal{E}$. In addition to this, Algorithm 5.1 requires as input an event sequence $\mathcal{S}$ from which contexts of occurrences of events are extracted, and a window width W within which events, that are taken into account when extracting the contexts, are supposed to occur. Instead of only one event sequence $\mathcal{S}$, the input of the algorithm can also consist of a set S of event sequences. The output of the algorithm are the sets of contexts for each event type in $\mathcal{EI}$. Computing a context $\mathsf{con_f}((e_i, t_i), \mathcal{S}, W)$ on Line 3 of the algorithm depends on whether the contexts are supposed to be sets of event types, or sequences of events.

An algorithm similar to Algorithm 5.1 could be used to extract contexts from an event type sequence S, or from a set S of event type sequences. In such a case, the input of the algorithms are a set $\mathcal{E}$ of all possible event types, and a set $\mathcal{EI}$ of interesting event types when $\mathcal{EI} \subseteq \mathcal{E}$, in addition to the event type sequence S or the set S of sequences. This algorithm also requires a context size K as input, instead of the window width W. The output of the algorithm are also the sets of contexts for each event type in the set $\mathcal{EI}$. And similarly to the case of event sequences, computing a context $\mathsf{con_f}(e_i, i, S, K)$ in the algorithm depends on whether the contexts are supposed to be sets of event types, or sequences of events.

Algorithm for computing centroid vector distances

Centroid vector distances d_{cev} between event types in a set $\mathcal{EI}$ are computed using Algorithm 5.2. The input of this algorithm are a set $\mathcal{E}$ of all possible event types (needed in computing the centroid vectors), a set $\mathcal{EI}$ of interesting event types, and sets $\mathsf{contexts}_{set}(A)$ of set contexts for all the event types $A \in \mathcal{EI}$. The output of the algorithm are all the pairwise distances d_{cev} between the event types in the set $\mathcal{EI}$.

Algorithm 5.2 Centroid vector distances d_{cev} between event types

Input: A set $\mathcal{E}$ of event types, a set $\mathcal{EI}$ of interesting event types, and sets $\mathsf{contexts}_{set}(A)$ for each $A \in \mathcal{EI}$.

Output: Pairwise centroid vector distances between the event types in the set $\mathcal{EI}$.

Method:

```
for all event types A ∈ EI do
    compute the centroid vector cev(A) of the set contexts_set(A);
od;
for all event type pairs (A, B) where A and B ∈ EI do
    calculate d_cev(A, B);
od;
output the pairwise centroid vector distances d_cev(A, B);
```

Algorithm for computing average minimum distances

Algorithm 5.3 is used for computing the average minimum distances d_{avm} between event types in a set $\mathcal{EI}$. The input of this algorithm are a set $\mathcal{EI}$ of interesting event types and sets $\mathsf{contexts}_{seq}(A)$ of sequence contexts for all the event types $A \in \mathcal{EI}$. The output of the algorithm are all the pairwise distances d_{avm} between the event types in the set $\mathcal{EI}$.

Algorithm 5.3 Average minimum distance d_{avm} between event types
Input: A set $\mathcal{EI}$ of interesting event types and sets $\mathsf{contexts}_{seq}(A)$ for each $A \in \mathcal{EI}$.
Output: Pairwise average minimum distances between the event types in the set $\mathcal{EI}$.
Method:
1. **for** all event type pairs (A, B) where A and $B \in \mathcal{EI}$ **do**
2. calculate $d_{\mathsf{avm}}(A, B)$;
3. **od**;
4. output the pairwise average minimum distances $d_{\mathsf{avm}}(A, B)$;

Complexity considerations

Using Algorithm 5.1 we can extract contexts of $|\mathcal{EI}|$ event types. The time required for extracting a context depends on the window width W, but also on how many events occur within it. Assuming that the most important factor is the length of the window width, extracting a context takes $O(W)$ time. In the case of event type sequences, the time required for extracting a context in turn depends on the context size K, and, therefore, it takes $O(K)$ time. The numbers of occurrences of interesting event types in the set $\mathcal{EI}$ vary a lot. If we denote by e the maximum number of occurrences of an interesting event type, i.e., $e = \max_{A \in \mathcal{EI}} \{ |\mathsf{contexts}_{\mathsf{f}}(A)| \}$, the time complexity of Algorithm 5.1 is at most $O(e\,|\mathcal{EI}|\,W)$. In the case of event type sequences, the time complexity of the corresponding algorithm is at most $O(e\,|\mathcal{EI}|\,K)$. Algorithm 5.1 needs space for all the event types in the set $\mathcal{EI}$ and the event sequence $\mathcal{S}$ (or the event type sequence S). Because the contexts of the event types in the set $\mathcal{EI}$ are first output when they all have been computed, the total space complexity of this algorithm is at most $O(|\mathcal{EI}| + |\mathcal{S}| + e\,|\mathcal{EI}|\,W)$. On the other hand, if the input sequence of the algorithm is an event type sequence S, the total space complexity of the algorithm is $O(|\mathcal{EI}| + |S| + e\,|\mathcal{EI}|\,K)$. If the input of the algorithm consists of a set S of sequences, the symbols $|\mathcal{S}|$ and $|S|$ above should be replaced by the space needed for the whole set S of sequences.

For calculating the centroid vector distances between event types, Algorithm 5.2 first computes a centroid vector for each set $\mathsf{contexts}_{set}(A)$ of contexts, where $A \in \mathcal{EI}$. If the maximum size of these sets $\mathsf{contexts}_{set}(A)$ is denoted by e as above, and there are $|\mathcal{E}|$ possible event types, then computing a centroid vector of a set $\mathsf{contexts}_{set}(A)$ takes $O(e\,|\mathcal{E}|)$ time, and when there are $|\mathcal{EI}|$ interesting event types, computing all the centroid vectors takes $O(|\mathcal{EI}|\,e\,|\mathcal{E}|)$ time. Assuming that computing the centroid vector distance d_{cev} between two event types takes $O(|\mathcal{E}|)$ time, computing the centroid vector distances d_{cev} between $\binom{|\mathcal{EI}|}{2}$ pairs of interesting event types takes $O(|\mathcal{EI}|^2\,|\mathcal{E}|)$ time. This means that the total time complexity of Algorithm 5.2 is $O(|\mathcal{EI}|\,e\,|\mathcal{E}| + |\mathcal{EI}|^2\,|\mathcal{E}|)$. Because the algorithm needs space for all the context vectors, the centroid vectors, and the pairwise centroid vector distances between the event types in the set $\mathcal{EI}$, the space complexity of Algorithm 5.2 is at most $O(|\mathcal{EI}|\,e\,c + |\mathcal{EI}|\,|\mathcal{E}| + |\mathcal{EI}|^2)$, where c is the maximum size of a context.

In order to compute the average minimum distance d_{avm} between two event types A and B in a set $\mathcal{EI}$, all the pairwise distances of sequences belonging to the sets $\mathsf{contexts}_{seq}(A)$ and $\mathsf{contexts}_{seq}(B)$ have to be computed. If n_{max} is the maximum length of sequence contexts, the computation of an edit distance between two sequence contexts takes at most $O(n_{max}^2)$ time and space, as was observed in Section 4.3. Assume that, when $A \in \mathcal{EI}$, the maximum size of the sets $\mathsf{contexts}_{seq}(A)$ is e as above. For determining the average minimum distance d_{avm} between two sets of sequence contexts we have to compute at most $\binom{e}{2}$ edit distances, and thus computing the average minimum distance d_{avm} between two event types takes $O(e^2\,n_{max}^2)$ time. In Algorithm 5.3 we compute the average minimum distances d_{avm} of $\binom{|\mathcal{EI}|}{2}$ pairs of interesting event types. Thus, the total time complexity of Algorithm 5.3 is $O(|\mathcal{EI}|^2\,e^2\,n_{max}^2)$. Algorithm 5.3 needs space for all the sequences in the sets of contexts, the edit distances between pairs of such sequences, and the average minimum distances between pairs of the event types in the set $\mathcal{EI}$. Therefore, the space complexity of this algorithm is $O(|\mathcal{EI}|\,e\,n_{max} + |\mathcal{EI}|^2\,e^2 + |\mathcal{EI}|^2)$. Because the set of interesting event types, and especially the sets of contexts, can be very large, computing the average minimum distances d_{avm} of event types can, therefore, lead to severe computational problems, as will be observed in the following section.

5.5 Experiments

In this section we present some results of experiments on similarity between event types. We describe the data sets used in these experiments in Section 5.5.1, and then study the results of the experiments in Section 5.5.2. The experiments were run under the Linux operating system using either a PC with 233 MHz Pentium processor and 64 MB main memory, a PC with 400 MHz Celeron processor and 128 MB main memory, a PC with 600 MHz Pentium III processor and 512 MB main memory, or a 275 MHz Alpha EV4 server with 512 MB main memory.

5.5.1 Data sets

The experiments on similarity between event types were made with both synthetic and real-life data. The real-life data sets were a telecommunication network alarm data, a protein sequence data, and a course enrollment data. All the data sets resided in flat text files.

Synthetic data

We constructed four synthetic event sequences for our experiments on similarity between event types. The exact process of generating these sequences is described in Appendix B. The numbers of possible event types and actual events in the four synthetic event sequences are given in Table 5.2. Each synthetic sequence contains two event types B and C so that the types of events preceding their occurrences are about the same, bar the effects of random variation. In the experiments with these synthetic event sequences we used all the possible event types as the set $\mathcal{EI}$ of interesting event types.

Telecommunication alarm data

We used the same telecommunication alarm sequence in our experiments on event type similarity as in the event sequence similarity experiments in Chapter 4. In this sequence there are occurrences of 287 alarm types. The length of the alarm sequence is 73 679 alarms. The numbers of occurrences of different alarm types in this set vary a great deal: from one to 12 186 occurrences.

From the set of 287 alarm types we selected several sets $\mathcal{EI}$ of interesting alarm types. However, here we present results on only two of these sets. The first of these sets consists of all 23 alarm types that occurred from

Event sequence	Event types	Events
$\mathcal{S}_1$	13	6 038
$\mathcal{S}_2$	23	29 974
$\mathcal{S}_3$	33	94 913
$\mathcal{S}_4$	43	142 433

Table 5.2: The number of event types and actual events in the four synthetic event sequences.

100 to 200 times in the whole alarm sequence. The second alarm type set, on the other hand, is a set of eight alarm types. These alarm types were chosen so that the numbers of their occurrences are of different magnitude. The actual numbers of the occurrences of these alarm types vary between 10 and 1 000 occurrences. The alarm types belonging to these two sets $\mathcal{EI}$ are given in Table 5.3.

Protein sequence data

The protein data set is obtained from the SWISS-PROT Protein Sequence Database [SWI99], whose 37th release contains information about 78 350 protein sequences. Each entry in the database has 23 attributes, e.g., an identification number, a general description of the sequence, and an actual protein sequence. The set of possible event types in this case of protein sequences consists of the twenty amino acids given in Table 5.1, and the three non-standard symbols: B, X and Z. When experimenting with the protein sequence data, we defined the set $\mathcal{EI}$ of interesting event types as the set of all the twenty standard amino acids, i.e., we excluded from the set $\mathcal{EI}$ the non-standard symbols B, Z and X.

To form our basic data set of protein sequences we first selected from the SWISS-PROT database 585 entries containing a word *hemoglobin* in their descriptions, and then from each entry the actual protein sequence. In the set of 585 protein sequences, there are 85 824 occurrences of amino acids and the other three symbols. The numbers of occurrences of the 23 different event types vary between 34 and 9 853, whereas the numbers of occurrences of the real amino acids vary between 960 and 9 853 occurrences. The lengths of the protein sequences, in turn, vary between 19 and 453 events. The mean number of events per a protein sequence is nearly 147.

Computing the average minimum distances d_{avm} between event types turned out to be very time-consuming. Therefore, for experiments on the average minimum distances d_{avm} between the amino acids, we chose ran-

Set of alarm types	Alarm types
23 alarm types	715, 1001, 1072, 1132, 1547, 1564, 1571, 2200, 2535, 2558, 2583, 2733, 2902, 2909, 2915, 7001, 7007, 7010, 7030, 7172, 7194, 7700, 7712
8 alarm types	1571, 1886, 1940, 2263, 2692, 7701, 7414, 9301

Table 5.3: The sets $\mathcal{EI}$ of the chosen 23 and 8 alarm types in the telecommunication alarm data.

domly from the set of 585 protein sequences one hundred protein sequences to our second data set of protein sequences. In this set there are occurrences of 14 282 amino acids and the three non-standard symbols. The numbers of occurrences of the 23 event types vary between 7 and 1 606, whereas the numbers of occurrences of the real amino acids vary between 129 and 1 606 occurrences. The lengths of the protein sequences in the second data set vary between 24 and 399 events. The average number of events per a protein sequence in this set is almost 143.

Course enrollment data

In the experiments on similarity between event types we used the same course enrollment data as in the attribute similarity experiments in Chapter 3. Recall that this data set contains information about 6 966 students at the Department of Computer Science at the University of Helsinki, and it was collected between 1989 and 1996.

The original data set was a sequence where each event describes a course enrollment of one student. If each course is thought to present an event type, the enrollments of each student can also be seen as a sequence of events. If the terms, when the enrollments are made, are taken into account, we can regard such a sequence as an event sequence, and when the enrollment terms are omitted, it can be viewed as an event type sequence[3]. The lengths of these enrollment sequences vary a lot, from one to a total of 33 course enrollments, with an average of close to five courses per student.

[3] In our experiments we only took into account the exact order of courses in which they were written into the data set. The courses that a student enrolls in during the same term are, however, ordered alphabetically by their course codes. Considering all the possible orders of the course enrollments would have been interesting, but it was still left for future study.

The number of different courses in this student enrollment data set is 173, and, as stated earlier, the courses are divided into three classes: basic, intermediate and advanced level courses. The numbers of occurrences of the courses vary from one to 3 902.

From the set of 173 courses we selected several sets $\mathcal{EI}$ of interesting courses, but here only present results on two of them. The first set of interesting courses contains 18 frequently occurring courses, for which the numbers of occurrences vary between 357 and 3 902. Of these 18 courses, nine are basic level courses and nine intermediate level courses. The courses in this set are given in Table 5.4. The time span of the data is seven years. During such a long time period the courses given at the department change. Therefore, in the set of the 18 courses some courses contain similar issues, for example, the course *C Language and Unix Programming Environment* from the beginning of the time period was later divided into two courses *Introduction to Unix* and *Programming in C.*

The other set $\mathcal{EI}$ of courses considered here is the set of nine advanced level courses already studied in Chapter 3. The courses in this set covering courses from the sections of computer software, information systems, and general orientation in computer science are given in Table 5.5. In this set the numbers of occurrences of courses vary between 84 and 342.

5.5.2 Results

In this section we first consider the centroid vector distances d_{cev} between event types in our test sets. After that we describe the average minimum distances d_{avm} between event types, and then compare these average minimum distances to the centroid vector distances d_{cev}. At the end of the section we also show some hierarchies of event types obtained by using the different similarity measures.

The distances between event types are computed using programs corresponding to the algorithms of Section 5.4. Like in Sections 3.5.2 and 4.4.2 where we describe results on attribute and event sequence similarity, respectively, we give some examples of the actual distance values here, but mostly study changes in the orders of distance values given by the measures.

Centroid vector distances with fixed values of K and W

We started our experiments by computing centroid vector distances d_{cev} between event types in the four synthetic event sequences. The window width W used in extracting the contexts of different event types was 5 time units. Figure 5.3 presents four histograms describing the

Course level	18 courses
basic	C Language and Unix Programming Environment, Computer Systems Organization, Fundamentals of ADP, Information Systems, Introduction to Unix, Programming in C, Programming (Pascal), Programming Project, Social Role of ADP
intermediate	Artificial Intelligence, Computer Graphics, Computers and Operating Systems, Data Communications, Database Systems I, Data Structures, Data Structures Project, Information Systems Project, Theory of Computation

Table 5.4: The set $\mathcal{EI}$ of 18 frequently occurring courses in the student enrollment data grouped by course level.

Section	9 courses
computer software	Compilers, Computer Networks, Distributed Operating Systems
information systems	Database Systems II, Object-Oriented Databases, User Interfaces
general orientation	Design and Analysis of Algorithms, Neural Networks, String Processing Algorithms

Table 5.5: The set $\mathcal{EI}$ of 9 advanced level courses in the student enrollment data grouped by section.

distributions of the centroid vector distances d_{cev} between the event types in these four event sequences. All these distributions roughly resemble the normal distribution. When looking at the exact distances, we noticed that in each case the most similar event types were B and C. This was an expected result, because the synthetic event sequences were generated so that they contained this pair of event types which have very similar sets of contexts. This means that by using the centroid vector distance measure we can truly recognize event types with similar contexts.

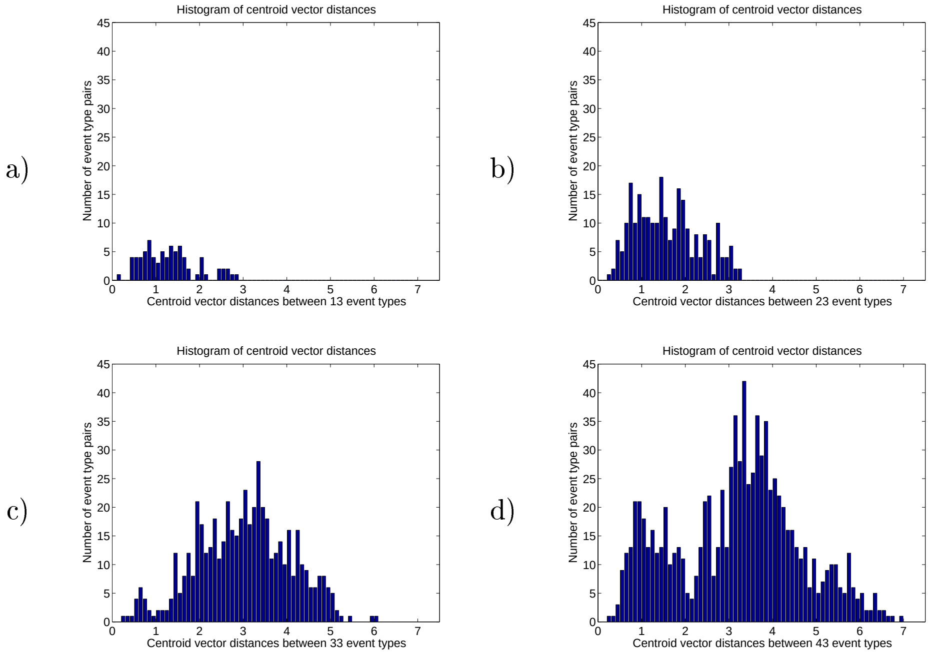


Figure 5.3: Distributions of centroid vector distances d_{cev} between the event types in the synthetic event sequence a) $\mathcal{S}_1$, b) $\mathcal{S}_2$, c) $\mathcal{S}_3$ and d) $\mathcal{S}_4$.

We then experimented with the set of 23 alarm types occurring from 100 to 200 times in the telecommunication alarm sequence. We extracted for each alarm type its set contexts using a window width W of 60 seconds, and computed the centroid vector distances d_{cev} between these 23 sets of contexts. Figure 5.4 presents a histogram of the distribution of these distances. From the histogram we can see that the absolute values of the centroid vector distances are quite high. This indicates that the sets of contexts of these alarm types, and thus the alarm types themselves, are not in general very similar. However, they are not extremely dissimilar either. The most similar pair in this set are the alarm types 2583 and 2733. These alarm types occur 181 and 102 times in the whole alarm sequence, respectively. On the other hand, the most dissimilar pair of the alarm types is the pair $(1564, 7172)$, where the alarm type 1564 occurs 167 times, and the alarm type 7172, in turn, 101 times in the whole alarm sequence. The number of occurrences does not seem to explain, why the first two alarm types are deemed similar and the other two very dissimilar. A more probable explanation is that the alarm types 2583 and 2733 belong to the same group of alarm types, whereas the alarm types 1564 and 7172 describe very

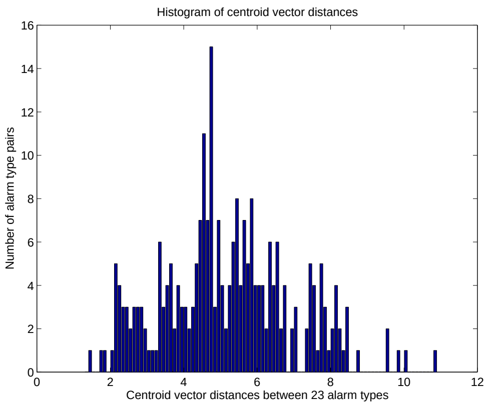


Figure 5.4: Distribution of centroid vector distances d_{cev} between the 23 alarm types with the window width $W = 60$.

different kinds of failures in the telecommunication network. The centroid vector distances d_{cev} also show that the alarm type 7172 is quite dissimilar to all the other 22 alarm types; its smallest centroid vector distance is to the alarm type 7010.

For the second experiment with the alarm data we chose some of the 23 alarm types, and considered one of the chosen alarm types at a time. We modified the alarm sequence so that every occurrence of the chosen alarm type A was independently changed to an event of the type A' with a probability of 0.5. Then, we extracted the set contexts for each of the now 24 alarm types with the window width $W = 60$, and computed the centroid vector distances d_{cev} between these sets of contexts. Our assumption was that the alarm types A and A' should be the most similar alarm type pair, similarly to the case of the event types B and C in the synthetic event sequences. This assumption, however, showed to be wrong with all the chosen alarm types. The reason for this is simply that the sets of contexts of the original alarm types were not very homogeneous. Therefore, the sets of the contexts of the alarm types A and A' could not be that, either.

We also studied the centroid vector distances d_{cev} between the twenty amino acids occurring in the set of protein sequences. In extracting the set contexts for each of the amino acids from the set of 585 event type sequences, we used a context size $K = 5$. Figure 5.5 presents a histogram of the centroid vector distances d_{cev} between the twenty amino acids. The distribution of the distances roughly resembles the normal distribution. Similarly to the case of the 23 alarm types, the centroid vector distances between amino acids are quite long, indicating that the sets of contexts of

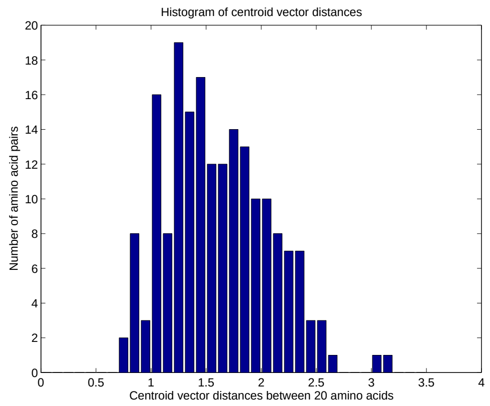


Figure 5.5: Distribution of centroid vector distances d_{cev} between the twenty amino acids occurring in the set of protein sequences with the context size $K = 5$.

different amino acids are not very similar to each other. The most similar pair of amino acids according to the centroid vector distance measure is the pair (*Glutamic acid*, *Lysine*), but the centroid vector distance between the amino acids *Glutamic acid* and *Leucine* is nearly as short. The most dissimilar amino acids according to this measure, on the other hand, are *Cysteine* and *Tryptophan*.

To better evaluate the centroid vector distances d_{cev} between amino acids, we compared them with the values of amino acid similarity given in two amino acid substitution matrices: the PAM 250 matrix [Pam99] and the BLOSUM 62 matrix [Blo99]. There are several PAM and BLOSUM matrices, but especially these two matrices are widely used in sequence alignments computed for protein database searches [Gus97]. The values in these matrices are based on the observed amino acid substitutions during evolution. These values are positive if a substitution of the amino acids considered has occurred by chance during evolution more often than expected, and negative if such a substitution has happened more seldom than expected. In general, the difference between PAM matrices and BLOSUM matrices is that PAM matrices are extrapolated from data obtained from very similar sequences, whereas BLOSUM matrices were developed explicitly to represent more distant, but still important relationships between the amino acids [Gus97].

Figure 5.6 presents two plots that describe how centroid vector distances between the amino acids are related to their PAM 250 similarities and their BLOSUM 62 similarities. In both cases the correlation between the values

a) b)

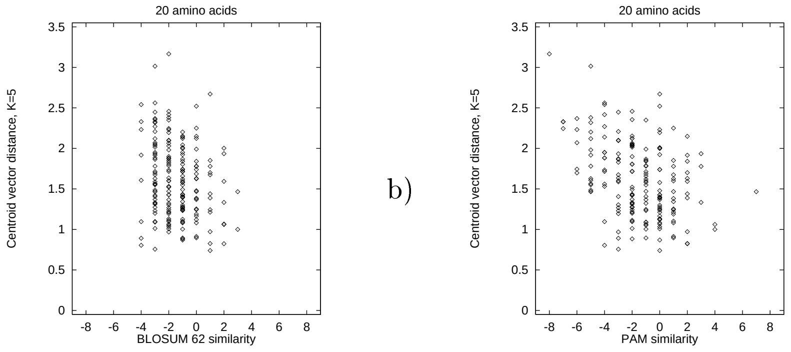


Figure 5.6: Comparison of centroid vector distances d_{cev} between the twenty amino acids with the context size $K = 5$ and the similarity values given in a) the BLOSUM 62 matrix and b) the PAM 250 matrix.

of the measures is slightly negative. This result is still natural, because the PAM 250 matrix and the BLOSUM 62 matrix give us information about similarities between the amino acids, whereas the centroid vector distances describe their distances. In other words, if all the measures would describe similarities between the amino acids, the correlation between them would be slightly positive. The distributions of the points in both of the plots in Figure 5.6 are quite wide, and even if several pairs of amino acids have exactly the same similarity value in the PAM 250 matrix, or in the BLOSUM 62 matrix, their centroid vector distances can be very different. The measures disagree on the most similar and the most dissimilar pair of the amino acids, and in general the orders of their values are also different. Therefore, it is clear that the measures describe the data differently.

As the last case, we computed centroid vector distances d_{cev} between the 18 frequently occurring courses in the student enrollment data. As mentioned in Section 5.5.1, a sequence of course enrollments can be seen either as an event type sequence or an event sequence. Thus, we extracted set contexts for each of the 18 courses from both types of sequences. When the sequences were treated as event type sequences, we used a context size $K = 5$ in extracting the set contexts, i.e., each context contains at the most five courses that the student has enrolled in before enrolling in the course considered. On the other hand, when the sequences were treated as event sequences, in selecting the set contexts we used a window width $W = 2$. This, in turn, means that each set context contains all the courses that the student has enrolled in during two terms at the most before enrolling in the course considered; note that of the courses the student enrolled in during the same term, only those whose course code in the alphabetical order is before the code of the considered were taken into account in extracting the set contexts. The reason for the choices of these parameter values is simple. The students at the department typically enroll in two or three courses per term, and thus, the set contexts with $W = 2$ are on average similar to the set contexts with $K = 5$.

We compared the centroid vector distances d_{cev} between the 18 courses computed using the two types of set contexts with each other. Figure 5.7 presents the relationship between these distances and shows that the distances have a clear positive linear correlation. The measures agree on the most similar pair of courses which is formed by the courses *Fundamentals of ADP* and *Programming Pascal.* This is a very natural result, because these courses are the first two courses recommended to be taken. Their set contexts are very often empty sets, or contain only the other course. Therefore, it is understandable that their sets of contexts are nearly the same. Many other pairs of the first year courses also have short centroid vector distances. On the other hand, the centroid vector distances between the first year courses and the courses that are supposed to be taken later during the studies are rather long, indicating very different sets of contexts, which is natural.

Despite the facts above, these two measures do not describe the relationships between the 18 courses exactly the same way: the orders of the distances are somewhat different. These measures disagree on the most dissimilar pair of courses, for example. With the context size $K = 5$ the courses *Social Role of ADP* and *Programming Project* have the longest centroid vector distances, whereas with the window width $W = 2$ the courses *Computer Systems Organization* and *Theory of Computation* are the most dissimilar pair of courses. This shows that the centroid vector distances d_{cev} between the courses vary depending on, whether we take into account the exact terms when the enrollments are made or not.

Intuitively, we can expect that two courses should be similar, if they are located approximately at the same stage of the curriculum. To find out how the centroid vector distances fulfill this expectation, we wanted to compare them to some kind of *background distances* of courses. Each course given at the department has a recommended term in which the department suggests it should be taken. Thus, we defined the background distance between two courses as the difference of the ordinals of their recommended terms. In our set of 18 courses the background distances have values from zero to seven terms. The comparisons of the background distances and the centroid vector distances d_{cev} are shown in Figure 5.8. In both cases there is a positive correlation between the background distances and the

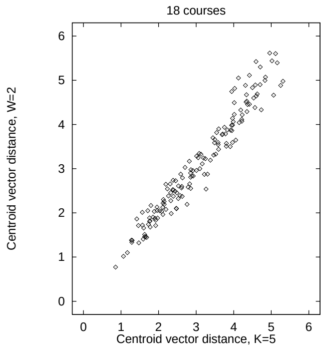


Figure 5.7: Comparison of centroid vector distances d_{cev} between the 18 courses, when the set contexts are extracted from the event type sequences with $K = 5$ and from the event sequences with $W = 2$.

a) b)

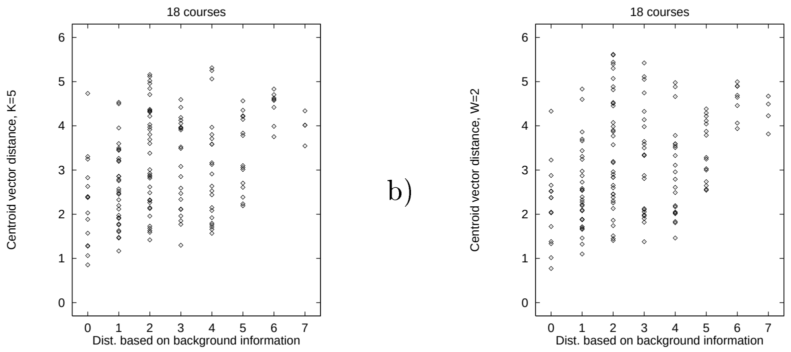


Figure 5.8: Comparison of background distances and centroid vector distances d_{cev} between the 18 courses, when the set contexts are extracted a) from the event type sequences with $K = 5$, or b) from the event sequences with $W = 2$.

centroid vector distances d_{cev}. This means that in a large perspective the centroid vector distances satisfy the intuition that two courses are similar, if they are supposed to be taken approximately at the same time during the studies.

The centroid vector distances, however, do not entirely agree with the background distances. For example, the centroid vector distance d_{cev} between the courses *Social Role of ADP* and *Introduction to Unix* is long, even though they are recommended to be taken during the same term. Similarly, the courses *Information Systems* and *Information Systems Project* have a long centroid vector distance, even though they are compulsory courses that are supposed to be taken during consecutive terms. This means that the set contexts of such courses vary a great deal, indicating that not all the students follow the recommended study plan. Because the data set was collected during seven years, the recommended study plan has also changed a few times, which may explain some of the variation in the set contexts.

Centroid vector distances with varying values of K and W

Our expectation was that, when the context size K or the window width W used in extracting the set contexts changes, the centroid vector distances d_{cev} between event types also vary. To find out, if this expectation is true, we selected some of our test sets of event types and computed the centroid vector distances d_{cev} for these event types with different window widths W or context sizes K.

We started these experiments with the set of eight alarm types occurring from 10 to 1 000 times in the telecommunication alarm sequence. For these alarm types we extracted set contexts using eight window widths W, and computed the centroid vector distances d_{cev} for each of these cases. The values of W used were $10, 20, 30, 60, 120, 300, 600$ and 1 200 seconds.

Figure 5.9 presents how the centroid vector distances d_{cev} between the eight alarm types vary when the window width W increases; Figure 5.9a shows the distances between the alarm type 2263 and the seven other alarm types, and Figure 5.9b the distances between the alarm type 9301 and the seven other alarm types. Similar results were also obtained with the other chosen alarm types. The alarm type 2263 occurred about 1 000 times in the whole alarm sequence, and the alarm type 9301 about 500 times. In both these cases, with the window width $W = 60$ already, the order of the centroid vector distances d_{cev} is rather stable; with longer time windows there are only slight changes in the orders of the distances. When we looked at the actual distance values, we could see that the centroid vector distances d_{cev} also indicate the differences in the numbers of occurrences of the alarm types. The alarm type 7414 occurs only 10 times in the whole alarm sequence, for example, whereas the alarm type 2263 occurs about 1 000 times, and the alarm type 9310 about 500 times. Thus, it is rather obvious that, if the numbers of occurrences of alarm types are of a different magnitude, especially with greater window widths, the sets of the contexts of these alarm types are different, and the centroid vector distances between them also become very long.

The same kind of experiments were also made with the protein sequence data. In extracting the set contexts for each of the twenty amino acids we

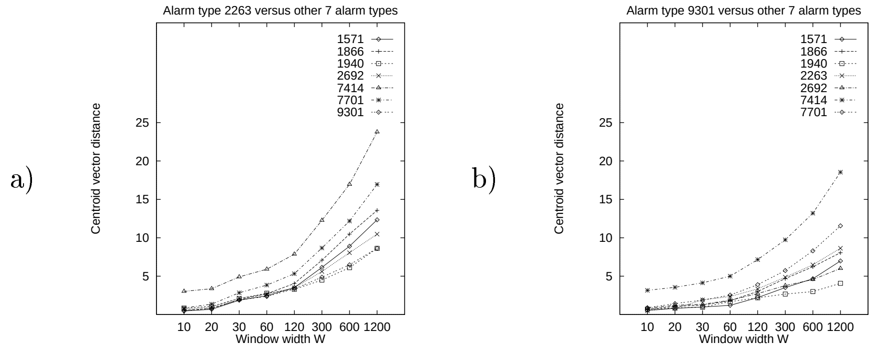


Figure 5.9: Comparison of centroid vector distances d_{cev} between a) the alarm type 2263, and b) the alarm type 9301 and the other alarm types with the chosen window widths W.

used seven context sizes K. The values of K used were $1, 2, 3, 4, 5, 10$ and 15. A comparison of the centroid vector distances d_{cev} between the amino acid *Alanine* and the other amino acids with these chosen context sizes is presented in Figure 5.10a, and the comparison of the centroid vector distances d_{cev} between the amino acid *Cysteine* and the other amino acids in Figure 5.10b. Very similar results would be obtained from comparisons of the other 18 amino acids. In this set of protein sequences the orders of the centroid vector distances d_{cev} do not become stable with any of the context sizes K used. This means that at least with small context sizes K, the choice of the context size depends very much on the situation considered. Typically the longest centroid vector distances d_{cev} were found between the amino acid *Cysteine* and the other amino acids, especially with the larger context sizes K. The explanation for this is the same as with the set of the eight alarm types: in the protein sequences, the amino acid *Cysteine* occurs only 960 times, whereas most of the other amino acids occur over 2 000 times. Thus, in this case the centroid vector distances d_{cev} also reflect the differences in the numbers of the occurrences of the amino acids.

We also studied the corresponding changes in the centroid vector distances d_{cev} between the nine advanced level courses in the student enrollment data set with different context sizes K and window widths W. In extracting the set contexts of these courses we used five context sizes K, when the sequences of course enrollments were seen as event type sequences, and five window widths W, when the sequences were regarded as event sequences. The parameter values in both cases were $1, 2, 3, 4$ and 5. Figure 5.11a presents how the centroid vector distances d_{cev} between the course *Design and Analysis of Algorithms* and the other eight courses

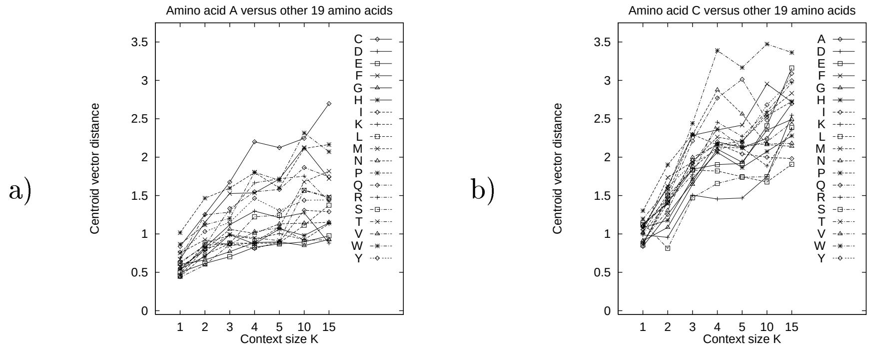


Figure 5.10: Comparison of centroid vector distances d_{cev} between a) the amino acid Alanine and b) the amino acid Cysteine and the other amino acids when the context size K is altered.

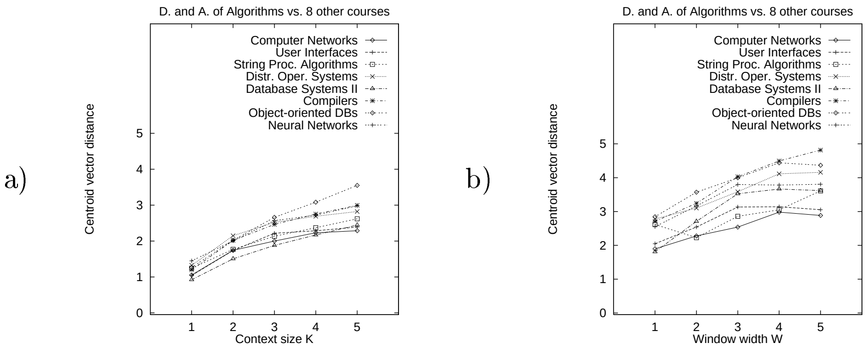


Figure 5.11: Comparison of centroid vector distances d_{cev} between the course *Design and Analysis of Algorithms* and the other courses when a) the context size K, and b) the window width W varies.

alter with the chosen context sizes K. The same kind of comparison of the centroid vector distances d_{cev} with the chosen window widths W is given in Figure 5.11b. The results of comparisons of the centroid vector distances d_{cev} for the other courses were very similar.

In general, the absolute centroid vector distances d_{cev} between the nine courses vary only a little with the different values of K and W. The comparisons also show that there is no context size K, or window width W after which the order of the centroid vector distances d_{cev} between the nine courses would become stable. An explanation for this is that the differences in the absolute distance values are rather small, especially in the case where the set contexts are extracted from the event type sequences. This, in turn, indicates that the sets of contexts of the courses are rather similar to each other. Unlike the alarm data and the protein data, the centroid vector distances d_{cev} between these courses do not at all reflect the numbers of the occurrences of the courses.

Average minimum distances with fixed values of K and W

We also wanted to see how the average minimum distance measure d_{avm} in practice succeeds in describing similarities between event types. Unfortunately, computing the average minimum distances d_{avm} showed out to be very complicated and time-consuming. An edit distance between two sequences can be computed reasonably quickly as shown in Chapter 4. However, the numbers of the occurrences of the chosen event types in the data sets used were typically high, i.e., hundreds or thousands of occurrences. In order to get the average minimum distances d_{avm} between two such event types, we have to compute all the pairwise edit distances between their sequence contexts, and these computations altogether take a long time. Therefore, we had to use, in most of our experiments on the average minimum distances d_{avm}, smaller sets $\mathcal{EI}$ of interesting event types, smaller values of the context size K and the window width W, and even smaller sets of original sequences. In computing the edit distances between two sequence contexts we used the unit operation costs with the parameter value $V = \frac{1}{W}$, and the alphabet-weighted operation costs with the parameter value $V = \frac{2 \cdot \mathsf{min\_w}}{W}$ (see Chapter 4).

We first computed the average minimum distances d_{avm} between the 13 event types occurring in the synthetic sequence $\mathcal{S}_1$. For each event type we extracted all its sequence contexts with the window width $W = 5$. Figure 5.12 presents the distributions of the average minimum distances d_{avm} between these event types when a) the unit operation costs, and b) the alphabet-weighted operation costs are used. In both cases the absolute distances are all less than 0.5. There are no remarkable differences between the distances computed using different operation costs. What surprised us first was that in both cases the most similar pair of event types was the pair (A_1, A_2), and not the pair (B, C) as in the case of the centroid vector distances. If we, however, consider the way the synthetic event sequences were generated (see Appendix B), it is quite clear that even if the set contexts of the event types B and C are similar, their sequence contexts do not have to be similar at all. Therefore, the results with the average minimum distances still reflect similarities between event types.

We also computed the average minimum distances d_{avm} between the 23 alarm types occurring in the telecommunication alarm sequence. Similarly

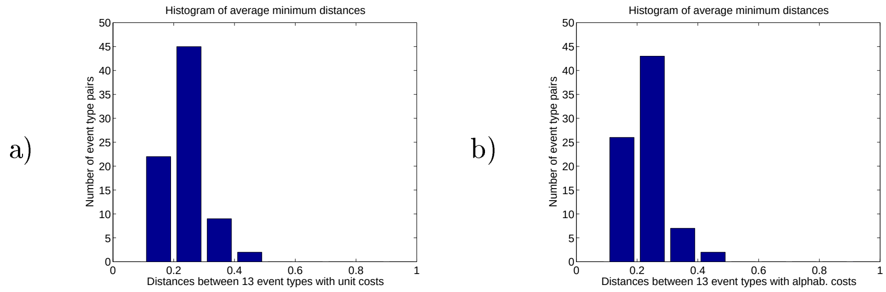


Figure 5.12: Distributions of average minimum distances d_{avm} between the 13 event types in the synthetic event sequence $\mathcal{S}_1$ when a) the unit operation costs and b) the alphabet-weighted operation costs are used.

to the case of the set contexts, we used the window width of 60 seconds for extracting the sequence contexts of these alarm types. The distributions of the average minimum distances d_{avm} between the 23 alarm types using both the unit and the alphabet-weighted operation costs are presented in Figure 5.13. The absolute distances are all more than 0.2, and both distributions are roughly normal. Comparison of the average minimum distances d_{avm} with different types of operation costs showed that the distances are positively linearly correlated, with slight differences in the absolute values. The most similar alarm types with the unit costs are the alarm types 2558 and 7001, whereas the second shortest distance with these costs is found to be between the alarm types 2583 and 2733. When the alphabet-weighted operation costs are used, the two most similar pairs of alarm types are the same, only the order of these pairs is different. With both types of operation costs, the most dissimilar pair of alarm types is the pair $(1564, 2200)$. In fact, the average minimum distances d_{avm} between the alarm type 1564 and the other alarm types are all very long, only with two alarm types its average minimum distance d_{avm} is less than 0.8.

Similar experiments on the average minimum distances d_{avm} were also made with the set of the twenty amino acids. For these experiments we extracted for each amino acid all its sequence contexts from the set of 100 protein sequences using the context size $K = 5$. We first compared the average minimum distances d_{avm} between the twenty amino acids obtained with the different types of operation costs, and observed that, similarly to the case of the synthetic event types and the chosen alarm types, the distances are clearly positively correlated, and even their absolute values are very close to each other. The distance measures also agree on the most similar pair of amino acids which is the pair (*Alanine*, *Serine*), whereas

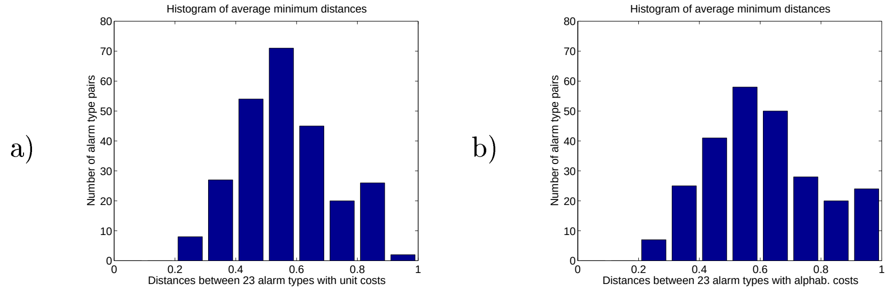


Figure 5.13: Distributions of average minimum distances d_{avm} between the 23 alarm types using a) the unit operation costs and b) the alphabet-weighted operation costs.

the two most dissimilar amino acids according to both the measures are the amino acids *Cysteine* and *Tryptophan*. The absolute distances in both cases were also very short, all less than 0.45.

We then compared the average minimum distances d_{avm} between the amino acids with the corresponding similarity values given in the BLOSUM 62 and the PAM 250 matrices. Comparisons of the average minimum distances d_{avm} computed using the alphabet-weighted operation costs and the amino acid substitution scores are shown in Figure 5.14. In both cases the average minimum distances d_{avm} are slightly negatively correlated with the similarity values in the matrices. The situation now is the same as with the centroid vector distances d_{cev}: if the values of the PAM and the BLOSUM matrices described distances between the amino acids, the correlations between the average minimum distances d_{avm} and the similarity values in the matrices would be positive. Therefore, we can say that the average minimum distances really describe similarities between amino acids.

As the last case, we studied the average minimum distances d_{avm} between the nine advanced level courses in the course enrollment data. In extracting the sequence contexts of these courses we used the context size $K = 5$ when the exact times of the course enrollments were omitted, and the window width $W = 2$ when they were taken into account. Figure 5.15 presents comparisons of the average minimum distances d_{avm} between the nine courses with these different sets of sequence contexts using different types of edit operation costs. As in all the previous experiments, the average minimum distances d_{avm} between the nine courses with different types of operation costs are also highly linearly correlated. The order of the average minimum distances d_{avm} with the different types of operation costs is still slightly different. In three of the cases the most similar pair of courses

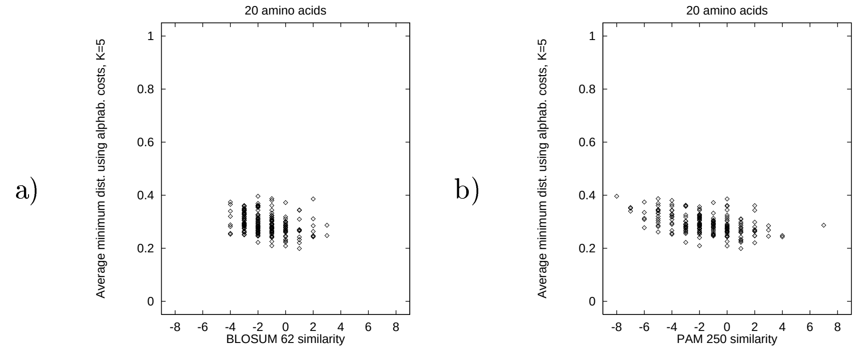


Figure 5.14: Comparison of average minimum distances d_{avm} between the twenty amino acids with the context size $K = 5$ and the similarity values given in a) the BLOSUM 62 matrix and b) the PAM 250 matrix.

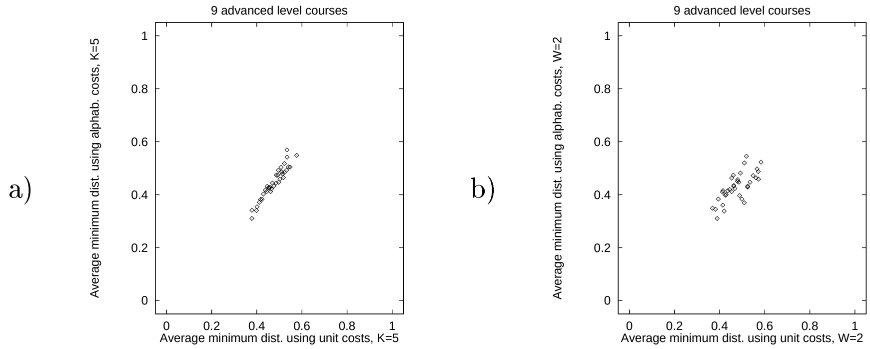


Figure 5.15: Comparison of average minimum distances d_{avm} between the nine advanced level courses with different operation costs using a) the context size $K = 5$ and b) the window width $W = 2$.

is the pair (*User Interfaces*, *Database Systems II*), but when the sequence contexts are extracted using the window width $W = 2$ and the average minimum distances are computed using the unit costs, the courses *Design and Analysis of Algorithms* and *Database Systems II* have the shortest distance. On the other hand, when the unit operation costs are used, the most dissimilar pair of courses with both the context size $K = 5$ and the window width $W = 2$ is the pair (*Object-Oriented Databases*, *Neural Networks*), whereas using the alphabet-weighted costs the courses *Computer Networks* and *Neural Networks* have the longest average minimum distance d_{avm} with both kinds of sequence contexts.

The average minimum distances d_{avm} between the nine advanced level courses were also compared with their background distances. These comparisons showed that the average minimum distances d_{avm} are not correlated with the background distances. This result can be explained by several facts. None of these nine courses is a compulsory course, and they all are courses from different sections of the department. Thus, only a small fraction of the students at the department enroll in each of these courses. On the other hand, they are advanced level courses, and when students are in that phase of their studies, they do not follow the recommended study plan as tightly as in the beginning of their studies. All this means that the sets of contexts of these courses are not particularly homogeneous. Therefore, it is only natural that the background distances give a very different view of the relationships between the courses from the average minimum distances d_{avm}, which actually describe the data in a better way.

Average minimum distances with varying values of K and W

Similarly to the case of centroid vector distances d_{cev}, we studied how the average minimum distances d_{avm} between event types vary when the context size K, or the window width W is altered. In this case also, the distances between two sequence contexts were computed using the unit operation costs with the parameter value $V = \frac{1}{W}$, and the alphabet-weighted operation costs with the parameter value $V = \frac{2 \cdot \mathsf{min\_w}}{W}$.

In the experiments with the set of the eight alarm types occurring in the telecommunication alarm sequence we used the window widths W of $10, 20, 30, 60, 120, 300, 600$ and 1 200 seconds in extracting the sequence contexts. Figure 5.16 presents how the average minimum distances d_{avm} between the alarm type 2263 and the other alarm types vary, when the window width W alters. The plots presenting the average minimum distances with the different types of operation costs are very similar. When the unit operation costs are used, the order of the average minimum distances d_{avm} becomes stable with the window width $W = 60$ already. Using the alphabet-weighted operation costs (Figure 5.16b), the order of the distances becomes stable somewhat later, with the window width $W = 120$.

The previous result is not, however, true with the other chosen alarm types. With them, the order of the average minimum distances d_{avm} does not really become stable at all, i.e., there is a small variation in the orders with every window width considered, both using the unit and the alphabet-weighted operation costs. This can be seen in Figure 5.17 presenting the average minimum distances d_{avm} between the alarm type 9301 and the other

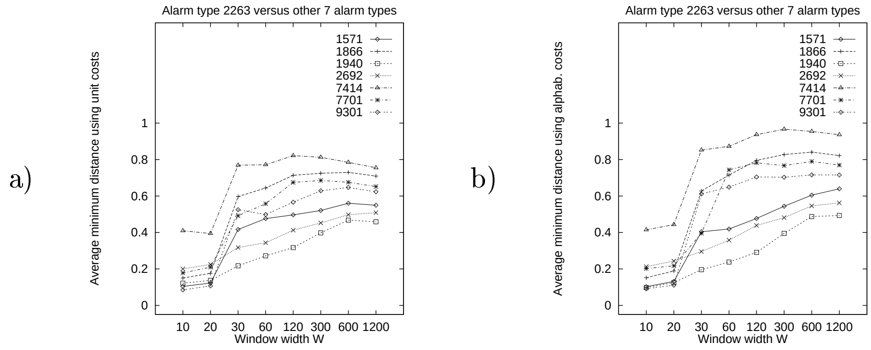


Figure 5.16: Comparison of average minimum distances d_{avm} between the alarm type 2263 and the other alarm types with different window widths W when a) the unit operation costs and b) the alphabet-weighted operation costs are used.

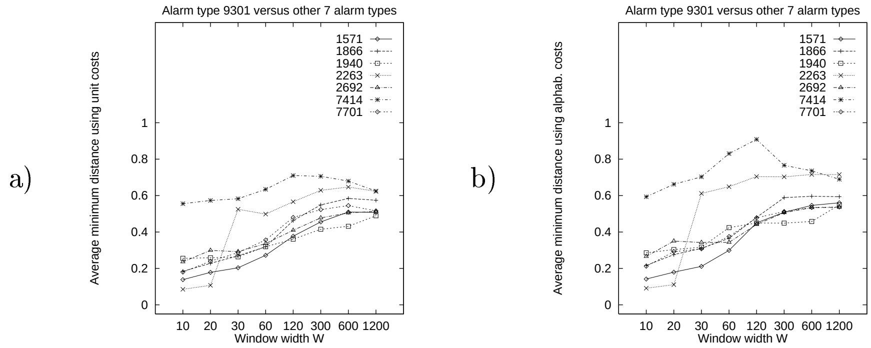
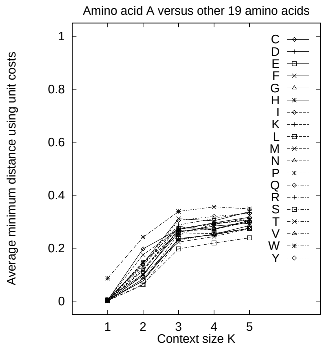


Figure 5.17: Comparison of average minimum distances d_{avm} between the alarm type 9301 and the other alarm types with different window widths W when a) the unit operation costs and b) the alphabet-weighted operation costs are used.

alarm types, for example. One reason for this difference between the alarm type 2263 and the other alarm types may be the different numbers of the occurrences of the alarm types. The alarm type 2263 occurs in the whole alarm sequence a total of 1 000 times being the most common of the eight alarm types, whereas the other alarm types occur 500 times in the whole sequence at the most. Another reason for the different behavior can be that, when the window width changes, the individual sequence contexts, and therefore, the sets of contexts of the alarm types other than the alarm type 2263 also change more differently with respect to each other than the set of contexts of the alarm type 2263. Note that, similarly to the case of

a) 

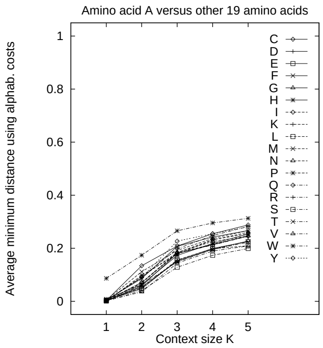
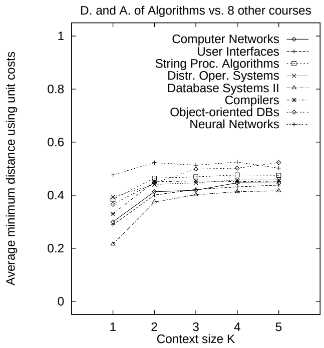
b) 

Figure 5.18: Comparison of average minimum distances d_{avm} between the amino acid *Alanine* and the other amino acids with different context sizes K when a) the unit operation costs and b) the alphabet-weighted operation costs are used.

the centroid vector distances d_{cev}, the alarm type 7414 typically has the longest distance to all the other alarm types.

We also made comparisons of the average minimum distances d_{avm} between the twenty amino acids with different context sizes K. The values of the context size K used in extracting the sequence context of the amino acids were $1, 2, 3, 4$ and 5; the greater context sizes 10 and 15 used in the case of the centroid vector distances d_{cev} had to be omitted because of computational limitations. Figure 5.18 presents the average minimum distances d_{avm} between the amino acid *Alanine* and the other amino acids with the chosen context sizes using a) the unit and b) the alphabet-weighted operation costs. The average minimum distances d_{avm} are all very close to each other; only the distances between *Alanine* and *Tryptophan* are slightly larger than the distances between *Alanine* and the other amino acids. These conclusions hold good for both types of edit operation costs. The order of the distances in both cases alters when the context size changes. This means that the order of the distances does not become stable with any of the chosen context sizes. The conclusions from the comparisons of the average minimum distances d_{avm} of the other amino acids were very similar.

Similar experiments on average minimum distances d_{avm} with different context sizes K and window widths W were also made in the course enrollment data for the set of the nine advanced level courses. In these experiments we used the same context sizes K and window widths W as in the case of the centroid vector distances d_{cev}, i.e., the values of K and W were both $1, 2, 3, 4$ and 5. The comparisons of the average minimum distances d_{avm} between the course *Design and Analysis of Algorithms* and

a) 

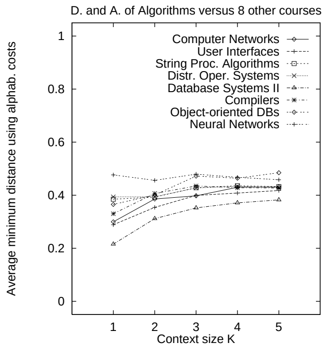
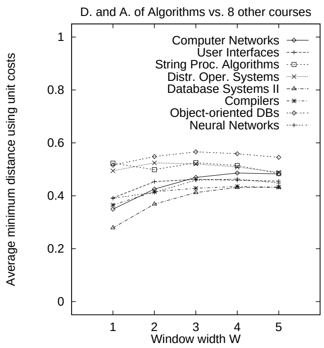
b) 

Figure 5.19: Comparison of average minimum distances d_{avm} between the course *Design and Analysis of Algorithms* and the other advanced level courses with different context sizes K when a) the unit operation costs and b) the alphabet-weighted operation costs are used.

a) 

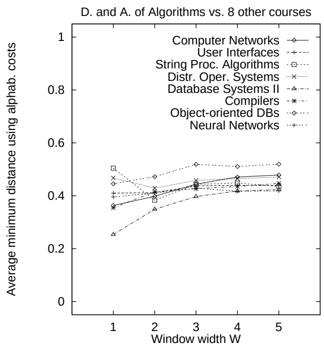
b) 

Figure 5.20: Comparison of average minimum distances d_{avm} between the course *Design and Analysis of Algorithms* and the other advanced level courses with different window widths W when a) the unit operation costs and b) the alphabet-weighted operation costs are used.

the other courses with the chosen context sizes K is shown in Figure 5.19, while corresponding comparisons with the chosen window widths W are presented in Figure 5.20. In both cases, the absolute average minimum distances d_{avm} are all rather close to each other. However, the order of the distances does not become quite stable in any of these cases with any of the parameter values considered; there is always some variation in the order when we move from one parameter value to another. Similar conclusions were also made from the comparisons of the average minimum distances d_{avm} of the other eight courses.

Centroid vector distances versus average minimum distances

The sets of contexts used in computing centroid vector distances d_{cev} and average minimum distances d_{avm} are different, and also the measures as such are different. We assumed that these two measures would describe similarities between event types in a different way, and thus, we wanted to compare these distances to see whether our assumption is true. The average minimum distances d_{avm} used in these experiments were computed using both the unit operation costs with $V = \frac{1}{W}$ and the alphabet-weighted operation costs with $V = \frac{2 \cdot \mathsf{min\_w}}{W}$.

We first studied the set of the 13 event types occurring in the synthetic event sequence $\mathcal{S}_1$. We used a window width $W = 5$ to extract the different types of contexts for each of these event types. Figure 5.21 presents the relationships between the centroid vector distances d_{cev} and the average minimum distances d_{avm} between these event types. The distributions of the average minimum distances d_{avm} computed using the unit operation costs (Figure 5.21a) and with the alphabet-weighted operation costs (Figure 5.21b) are rather similar, and the relationship between the centroid vector distances d_{cev} and the average minimum distances d_{avm} is positive in both cases. Nevertheless, the most similar pair of event types according to the centroid vector distance measure d_{cev} is the pair (B, C), but the average minimum distance measure d_{avm} finds the event types A_1 and A_2 most similar. Similarly, the measures disagree on the most dissimilar pair of event types. In general the orders of these distances also alter, and thus, it is clear that these measures give a different view of the similarities between these 13 event types.

We also compared the centroid vector distances d_{cev} and the average minimum distances d_{avm} between the 23 alarm types in the telecommunication alarm sequence. In these experiments, the contexts of the alarm types were extracted using the window width W of 60 seconds. Figure 5.22 presents the comparison of these distances when a) the unit operation costs, and b) the alphabet-weighted operation costs were used in computing the average minimum distances d_{avm}. The distributions of the points in these plots are slightly dissimilar, indicating the differences in the average minimum distances d_{avm} with the different type of operation costs. The differences are, however, small, and thus, both the average minimum distance measures d_{avm} describe the distances between the 23 alarm types similarly, as seen in our earlier experiments. In both cases, the centroid vector distances d_{cev} and the average minimum distances d_{avm} are positively correlated. When the alphabet-weighted operation costs are used, the measures

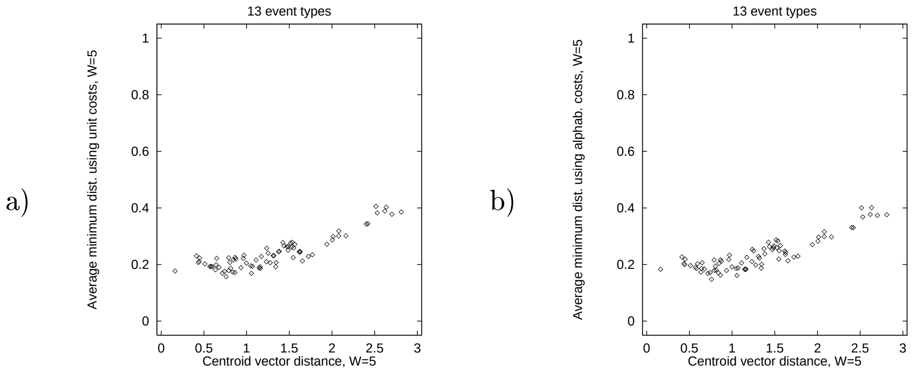


Figure 5.21: Comparisons of distances d_{cev} and d_{avm} between the 13 event types in the synthetic event sequence $\mathcal{S}_1$ when a) the unit costs and b) the alphabet-weighted costs were used in computing the average minimum distances d_{avm}.

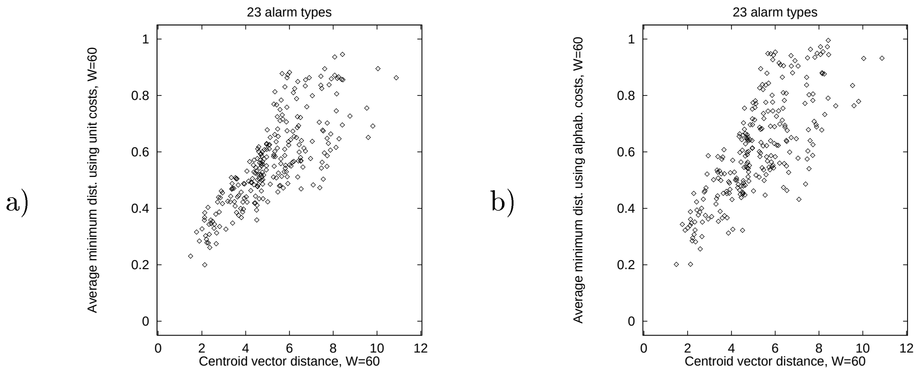


Figure 5.22: Comparisons of distances d_{cev} and d_{avm} between the 23 alarm types when a) the unit costs and b) the alphabet-weighted costs are used in computing the average minimum distances d_{avm}.

d_{cev} and d_{avm} agree on the most similar pair of alarm types, but disagree on what are the most dissimilar alarm types. On the other hand, with the unit operation costs, the measures d_{cev} and d_{avm} agree neither on the most similar nor on the most dissimilar pair of the alarm types. Because the orders of the other values of the measures vary as well, in this case the centroid vector distance measure d_{cev} and the average minimum distance measure d_{avm} also describe the similarities between the chosen event types differently.

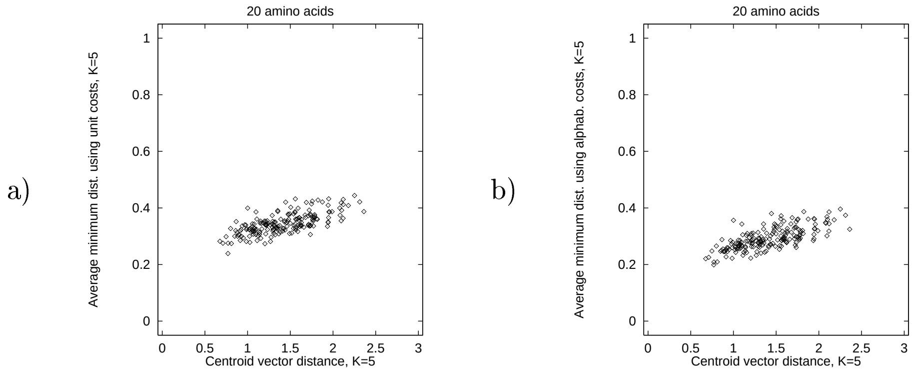


Figure 5.23: Comparisons of distances d_{cev} and d_{avm} between the twenty amino acids when a) the unit costs and b) the alphabet-weighted costs are used in computing the average minimum distances d_{avm} distances.

After that we focused on the distances between the twenty amino acids in the set of 100 protein sequences. We extracted the contexts of amino acids from these sequences using the context size $K = 5$. The comparisons of the centroid vector distances d_{cev} and the average minimum distances d_{avm} between the amino acids are shown in Figure 5.23, when a) the unit and b) the alphabet-weighted operation costs were used in computing the average minimum distances d_{avm}. These two plots are very much alike. This result is not surprising, because, as stated earlier in this section, the average minimum distances d_{avm} with different operation costs are very close to each other. From the plots we can also see that the centroid vector distances d_{cev} and the average minimum distances d_{avm} are positively correlated, but that the orders of the distances are not exactly the same. This result is also indicated by the fact that these measures disagree both on the most similar and the most dissimilar pair of amino acids. Hence, once again we can say that these measures give us a different view of the similarities between event types.

As the last case of these experiments we considered distances between the nine advanced level courses in the course enrollment data set. For these courses we extracted both set and sequence contexts using the context size $K = 5$ and the window width $W = 2$. Because the average minimum distances d_{avm} between these courses with the unit and the alphabet-weighted operation costs are very similar, regardless of whether the contexts are extracted from event type sequences or event sequences, Figure 5.24 presents only comparisons of the centroid vector distances d_{cev} and the average minimum distances d_{avm} with the alphabet-weighted operation costs. The dis-

a) b)

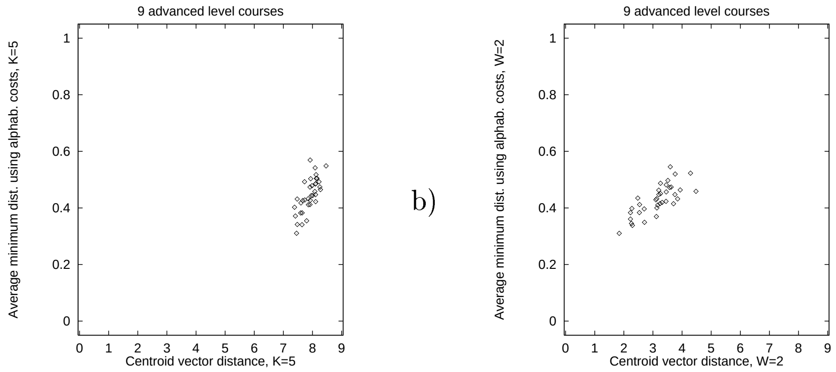


Figure 5.24: Comparisons of distances d_{cev} and d_{avm} between the nine advanced level courses with a) the context size $K = 5$ and b) the window width $W = 2$, when the alphabet-weighted costs are used in computing the pairwise edit distances needed for d_{avm} distances.

tribution of the points in the two plots are very different, indicating the differences in the centroid vector distances d_{cev}, when the set contexts are extracted using either the context size $K = 5$, or the window width $W = 2$. In both cases, the centroid vector distances d_{cev} are positively related to the average minimum distances d_{avm}. However, the orders of the distances given by the various measures in general are different. This means that in this case the centroid vector distance measure d_{cev} and the average minimum distance measure d_{avm} also describe the similarities between the chosen event types in a different way.

Hierarchies of event types

Similarities between event types as such provide us important information about the data. These similarities can also be used in defining similarity between event sequences, if we allow events to be substituted with events of a similar type. In such a case using a hierarchy of event types in addition to their actual similarity values may be useful. Such a hierarchy of event types can be built, for example, using the standard agglomerative hierarchical clustering [And73, JD88, KR90]. In the same way as we constructed clustering trees of binary attributes and sequences of events, we studied the clustering trees for the different sets of event types. In this case also, we used the three hierarchical clustering methods presented in Appendix A, i.e., the single, the complete, and the average linkage methods. In the following we give some examples of event type hierarchies constructed for the sets of the eight alarm types and the nine advanced level courses.

Figure 5.25 presents two clustering trees of the eight alarm types occurring in the telecommunication alarm sequence. Both the clustering trees were produced using the single linkage method, and they are both based on the centroid vector distances d_{cev} between the alarm types. The centroid vector distances d_{cev} for Figure 5.25a were computed using the set contexts of the alarm types with the window width $W = 60$, and for Figure 5.25b the set contexts with the window width $W = 600$. As seen earlier in this section, changing the window width W alters the order of distances which results in different clustering trees. Note that the clustering tree in Figure 5.25b is a typical example of the chaining effect of the single linkage method.

In our experiments with the centroid vector distances d_{cev} we also found that different kinds of similarity notions between event types can be obtained depending on whether the set contexts of the event types are extracted from event sequences or from event type sequences. Therefore, the clustering trees of these event types based on these different similarity measures should be different. This conclusion is supported by the event type hierarchies in Figure 5.26. The trees present the clustering trees of the nine advanced level courses based on their centroid vector distances d_{cev} when the set contexts of these event types are extracted a) from event type sequences using the context size $K = 5$, and b) from event sequences using the window width $W = 2$. In neither clustering tree are the courses divided into groups corresponding to the sections at the Department of Computer Science. The clustering trees are also different from the clustering trees of the same courses based on the internal and external measures of attribute similarity considered in Section 3.5.2.

Figure 5.27 presents six clustering trees of the eight alarm types occurring in the telecommunication alarm sequence, based on the average minimum distances d_{avm} between these alarm types. The window width W used in extracting the sequence contexts of the alarm types was 60 seconds, and in computing the distances both the unit and the alphabet-weighted operation costs were used. In producing the trees in Figures 5.27a and 5.27d we used the single linkage method, the trees in Figures 5.27b and 5.27e the complete linkage method, and the trees in Figures 5.27c and 5.27f the average linkage method. The clustering trees produced with the three methods are not particularly different. On the other hand, the differences between the clustering trees based on the distances with the different operation costs are even smaller than the differences with the various clustering methods.

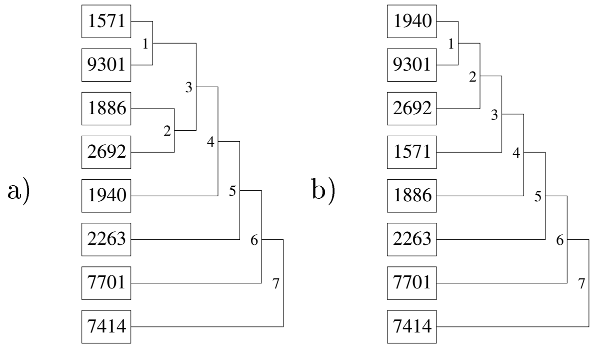


Figure 5.25: Clustering trees of the eight alarm types produced with the single linkage method when the centroid vector distances d_{cev} are used, and the window width is a) $W = 60$ and b) $W = 600$.

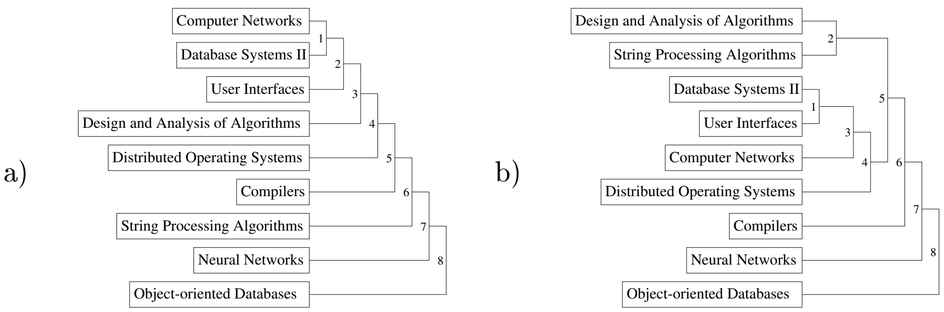


Figure 5.26: Clustering trees of the nine advanced level courses produced with the single linkage method when the centroid vector distances d_{cev} are used, and a) the context size is $K = 5$ and b) the window width is $W = 2$.

What is common to all the six clustering trees is that there are always three pairs of alarm types, i.e., the pairs $(1571, 9301)$, $(1886, 2692)$ and $(1940, 2263)$, that are grouped together. All these clustering trees are also different from the trees in Figure 5.25.

As in the case of attribute hierarchies (Section 3.5.2) and in the case of hierarchies of sequences (Section 4.4.2), the clustering trees presented here also indicate that when we use different similarity measures, we typically obtain different clustering trees, even if we use the same clustering method. On the other hand, if we consider the clustering trees obtained with a certain similarity measure, but vary the clustering method, the clustering trees are typically dissimilar. Still, in some cases the clustering trees

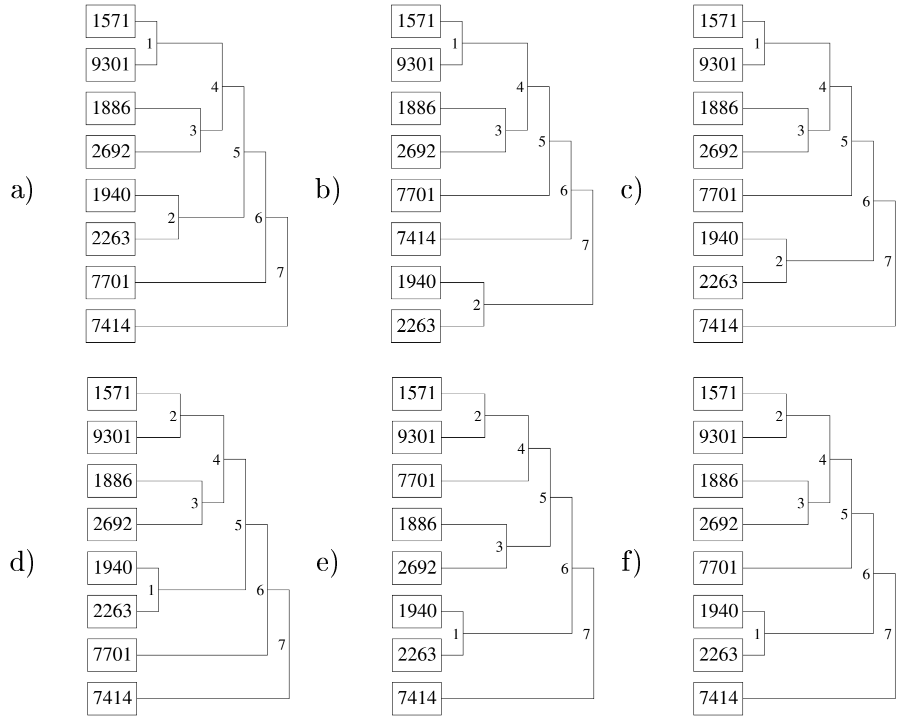


Figure 5.27: Clustering trees of the eight alarm types produced with the single (left column), the complete (center column), and the average (right column) linkage methods, when the window width is $W = 60$, and the average minimum distances d_{avm} with the unit costs (upper row) and the alphabet-weighted costs (lower row) are used.

produced using either different similarity measures or different clustering methods can be exactly the same. Note that even though we represented here only clustering trees of some of the sets of interesting event types, all these conclusions hold good for the other sets of interesting event types as well.

5.6 Discussion

In this chapter we have discussed different ways of defining similarity between event types occurring in sequences. The basic idea behind our approach was that two event types are similar, if they occur in similar contexts in sequences. This means that we must find the contexts of event types, and define similarity between two event types based on the similarity, or the distance between their sets of contexts. We considered two basic ways of defining the contexts: set contexts and sequence contexts. We then showed how to define the distance between two sets of set contexts as a centroid vector distance, and the distance between two sets of sequence contexts as an average minimum distance.

The results of our experiments on event type similarity were presented in Section 5.5. In the experiments we used both synthetic and real-life data sets. The results of these experiments show that the centroid vector distance and the average minimum distance describe the similarities between event types in different ways. This does not only indicate the differences between the measures as such, but the difference in the type of the contexts, as well. On the other hand, the values of the context size K and the window width W used in extracting the contexts also turned out to influence the similarities between event types. In contrast, the operation costs used in computing the average minimum distances did not seem to have any particular influence on the distances. This was a bit surprising, because in Section 4.4 the type of the edit operation costs was found to have a remarkable influence on the edit distances between sequences of events. The explanation to the completely opposite result of the previous section is, however, rather simple: taking the average of the minimum edit distances between the sequence contexts makes those differences disappear.

What similarity notion, then, should we use in each case? That is a very difficult question to answer. Depending on the choice of the type of contexts and the values of the context size and the window width, needed in extracting the contexts, we can obtain very different notions of event type similarity. Developing alternative distance measures for the centroid vector distance and the average minimum distance would also make the choice of the proper similarity measure more difficult. In general, we must either have enough domain knowledge to be able to select the best possible similarity measure for each case, or we have to select such a measure and the proper parameter values experimentally.

Chapter 6

Conclusions

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

We started in Chapter 2 by considering 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 distance measures to have. A measure that can be used for defining a distance between objects should be a metric, a pseudometric, or at least a semimetric. It should also be easy and efficient to compute. Furthermore, such a measure should be natural, and it should capture the intuitive similarity between objects.

Then, in Chapter 3, we presented various ways of defining similarity between binary attributes. We started by considering 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 often useful but, unfortunately, they cannot reflect all important types of similarity. Therefore, we moved on to discuss other types of attribute similarity measures. We introduced an external similarity measure that determines the distance between two attributes by considering the values of a selected set of probe attributes in the same relation.

Behavior of both internal and external measures of attribute similarity were demonstrated on two real-life data sets: the Reuters-21578 newswire data and the course enrollment data from 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. Using various probe sets also gave different similarity notions. This, however, is as it should be: the probe set defines the point of view from which similarity between attributes is judged.

After that, in Chapter 4, we studied how to determine similarity between two event sequences. Our main intuition in defining similarity between sequences of events is that it should reflect the amount of work that is needed to convert one event sequence into another. We formalized this notion as the edit distance between sequences, and showed that such a measure can be efficiently computed using dynamic programming.

We gave experimental results for event sequence similarity on two real-life data sets: a telecommunication alarm sequence and a log of WWW page requests. These results showed that our definition of the edit distance between sequences produces an intuitively appropriate notion of similarity. We also studied what the influence of associating different costs to the edit operations used in transformations is, and found that with different types of operation costs we get different notions of similarity. Moreover, the window width that describes the time period during which the events in the sequences are supposed to occur, was found to have a noteworthy effect on the distances between sequences.

Finally, in Chapter 5 we considered how similarity between event types occurring in sequences could be defined. The intuitive idea behind our definition of similarity between event types is that two event types are similar if they occur in similar contexts. First we studied two main ways of extracting contexts of occurrences of event types: set contexts and sequence contexts. We then gave two measures for defining similarity between event types. Of these, the centroid vector distance describes the distance between two event types when the contexts are sets of event types, and the average minimum distance when the contexts are event sequences or event type sequences.

In our experiments on event type similarity we used four synthetic event sequences and three real-life data sets: a telecommunication alarm sequence, a set of protein sequences, and a set of course enrollment data. These experiments showed that with different kinds of contexts we get different kinds of similarity notions. The context size and the window width used in extracting the contexts were also found to have a noticeable influence on the distances between event types. In general, these experiments showed that our measures can produce intuitively appealing results, and that they can locate similarities in various types of sequences.

Many interesting problems still 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. The influence of the proposed variations of the external measure could also be worthwhile to examine. Furthermore, more experiments should be made to determine the usability of external distances in various application domains. The use of attribute hierarchies in rule discovery and extensions of the external measure for a distance between attribute values are also worth investigating.

The event sequence similarity should also be developed further. An important extension would be to include to the set of edit operations a substitution of events, and then to examine how this change influences the distances between sequences. This extension would mean that we make use of event type similarity in determining similarity between event sequences. In our experiments we considered the events that had the same occurrence times in the sequences only in the exact order they were written in the database. However, in the future the influence of all the possible orders of such events on the edit distances between sequences should be examined. Further experimentation is also needed to determine the usability of edit distance in various application domains. Finally, considering other types of measures than edit distance in defining similarity between event sequences might be useful.

There is also a great deal of work to be done in the area of event type similarity. For one thing, various types of contexts of events could be considered. One of these types are the two-sided contexts of events, where events occurring both before and after the interesting event are taken into account. On the other hand, giving a weight to each event considered in extracting a context of an event, so that the weight indicates how close these events are to each other, might lead to interesting results. Studying the behavior of our measures in other application areas might also be useful. Similarly to the case of event sequence similarity, the influence of different orders of events that have the exactly same occurrence time in the sequences to the distances between event types should be studied. Furthermore, alternative measures for event type similarity should be considered. Because computing the average minimum distance between event types turned out to be extremely time-consuming, especially more efficient ways of defining similarity between sets of sequence contexts should be searched for in the future.

References

[ABKS98] M. Ankerst, B. Braunmüller, H.-P. Kriegel, and T. Seidl. Improving adaptable similarity query processing by using approximations. In A. Gupta, O. Shmueli, and J. Widom, editors, *Proceedings of the 24th International Conference on Very Large Data Bases (VLDB'98)*, pages 206 – 217, New York, NY, USA, August 1998. Morgan Kaufmann.

[AFS93] R. Agrawal, C. Faloutsos, and A. Swami. Efficient 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.

[BFG99] K. P. Bennett, U. Fayyad, and D. Geiger. Density-based indexing for approximate nearest-neighbor queries. In S. Chaudhuri and D. Madigan, editors, *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining (KDD'99)*, pages 233 – 243, San Diego, CA, USA, 1999. ACM.

[Blo99] Blosum62.cmp. The Wisconsin package (trademark), the GCG Manual, Version 10, Appendix VII.
http://ir.ucdavis.edu/GCGhelp/appendix_vii.html, 1999.

[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. Gunopulos, 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.

[EM97] T. Eiter and H. Mannila. Distance measures for point sets and their computation. *Acta Informatica*, 34(2):109 – 133, 1997.

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

[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.

[GIM99] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In M. Atkinson, M. E. Orlowska, P. Valduriez, S. Zdonik, and M. Brodie, editors, *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB'99)*, pages 518 – 529, Edinburgh, Scotland, UK, September 1999. Morgan Kaufmann.

[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.

[GRS99] M. N. Garofalakis, R. Rastogi, and K. Shim. SPIRIT: Sequential pattern mining with regular expression constraints. In M. Atkinson, M. E. Orlowska, P. Valduriez, S. Zdonik, and M. Brodie, editors, *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB'99)*, pages 223 – 234, Edinburgh, Scotland, UK, September 1999. Morgan Kaufmann.

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

[GWS98] V. Guralnik, D. Wijesekera, and J. Srivastava. Pattern directed mining of sequence data. 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 51 – 57, New York, NY, USA, 1998. AAAI Press.

[Ham50] R. W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29:147 – 160, 1950.

[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.

[HF95] J. Han and Y. Fu. Discovery of multiple-level association rules from large 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 420 – 431, Zürich, Switzerland, 1995. Morgan Kaufmann.

[HK90] D. Huttenlocher and K. Kedem. Computing the minimum Hausdorff distance for point sets under translation. In *Proceedings of the Sixth Annual Symposium on Computational Geometry*, pages 340 – 349, Berkeley, CA, USA, June 1990. ACM.

[HKK92] D. Huttenlocher, K. Kedem, and J. Kleinberg. On dynamic Voronoi diagrams and the minimum Hausdorff distance for point sets under Euclidean motion in the plane. In *Proceedings of the Eighth Annual Symposium on Computational Geometry*, pages 110 – 119, Berlin, Germany, June 1992. ACM.

[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 Computer Society Press.

[HR92] D. Huttenlocher and W. Rucklidge. A multi-resolution technique for comparing images using the Hausdorff distance. Technical Report TR 92-1321, Department of Computer Science, Cornell University, Ithaca, NY, USA, December 1992.

[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.

[KE98] Y. Karov and S. Edelman. Similarity-based word sense disambiguation. *Computational Linguistics*, 24(1):41 – 59, March 1998.

[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 (ALT'93)*, pages 1 – 18, Tokyo, Japan, nov 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.

[MM99] H. Mannila and P. Moen. Similarity between event types in sequences. In M. Mohania and A. M. Tjoa, editors, *Proceedings of the First International Conference on Data Warehousing and Knowledge Discovery (DaWaK'99)*, pages 271 – 280, Florence, Italy, August - September 1999. Springer-Verlag.

[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. In E. Simoudis, J. Han, and U. M. Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pages 146 – 151, Portland, OR, USA, August 1996. AAAI Press.

[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.

[Pam99] Pam250.cmp. The Wisconsin package (trademark), the GCG Manual, Version 10, Appendix VII. http://ir.ucdavis.edu/GCGhelp/appendix_vii.html, 1999.

[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.

[SWI99] The SWISS-PROT Protein Sequence Database, Release 37.
http://www.expasy.ch/sprot/ and
ftp://ftp.expasy.ch/databases/swiss-prot, 1999.

[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.

[Tve77] A. Tversky. Features of similarity. *Psychological Review*, 84(4):327 – 352, July 1977.

[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.

[WH98] G. M. Weiss and H. Hirsh. Learning to predict rare events in event sequences. 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 359 – 363, New York, NY, USA, August 1998. AAAI Press.

[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.

[Zha99] W. Zhang. A region-based learning approach to discovering temporal structures in data. In I. Bratko and S. Džeroski, editors, *Proceedings of the Sixteenth International Conference on Machine Learning (ICML'99)*, pages 484 – 492, Bled, Slovenia, June 1999. Morgan Kaufmann.

Appendix A

Hierarchical clustering

Discovering groups in data is an important problem in many application areas. In analyzing market basket data, for example, it is interesting to find market segments, i.e., groups of customers with similar needs, or to find groups of similar products. 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).

Research in the field of clustering has been extensive, and many different methods for grouping data have been developed; see [And73, JD88, KR90] for overviews of cluster analysis. The main goals of every clustering method are to find, for the given set of objects, a set of clusters where objects within each cluster are similar and objects in different clusters are very dissimilar to each other. Each of the clustering methods, however, describes the data from one point of view. This means that different methods may produce different kinds of clusterings. Because a clustering produced by one method may be satisfactory for one part of the data, and another method for some other part, it may in many cases 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.

One group of widely used clustering techniques is the *hierarchical clustering* methods. These hierarchical methods find partitions of objects such that each cluster contains at least one object and each object belongs to exactly one cluster, i.e., clusters are disjoint. Formally, the clusterings discovered by these methods can be defined as follows.

Definition A.1 Let $\mathbf{O} = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$ be a set of objects. A *clustering* $\mathcal{C}$ of the object set $\mathbf{O}$ 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 that contains only one object is called a *singleton cluster*. □

Instead of one single partition of the given objects, the hierarchical clustering methods construct a sequence of clusterings. Such a sequence of clusterings is often given as a *clustering tree*, also called a *dendrogram* [DJ76, DH73]. In such a tree, leaves represent the individual objects and internal nodes the clusters. An example of a clustering tree is shown in Figure A.1.

There are two kinds of hierarchical clustering techniques: *agglomerative* and *divisive*. The 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 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 A.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 to as *bottom-up* and the divisive methods as *top-down* hierarchical clustering. A common algorithm for agglomerative hierarchical clustering is given as Algorithm A.1.

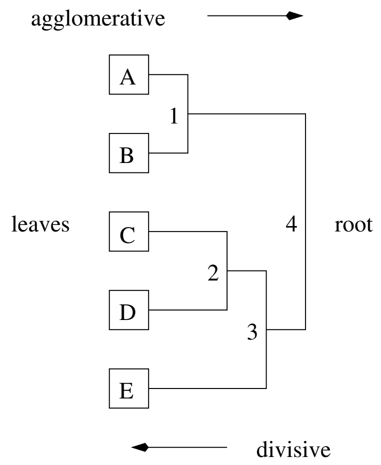


Figure A.1: An example of a clustering tree.

Algorithm A.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 $\mathbf{O}$.
Method:

1. $\mathcal{C}_0 =$ the trivial clustering of n objects in the input set $\mathbf{O}$;
2.   **for** $k = 1$ to $|\mathbf{O}| - 1$ **do**
3.   find $c_i, c_j \in \mathcal{C}_{k-1}$ so that the distance $d(c_i, c_j)$ is shortest;
4. $\mathcal{C}_k = (\mathcal{C}_{k-1} \setminus \{c_i, c_j\}) \cup (c_i \cup c_j)$;
5.   compute the distance $d(c_i, c_j)\ \forall\ c_i, c_j \in \mathcal{C}_k$;
6.   **od**;
7. output $\mathcal{C}_0, \ldots, \mathcal{C}_{n-1}$;

Algorithm A.1 gets as input a finite set $\mathbf{O}$ of n objects and a matrix of pairwise distances between these objects. This means that executing the clustering algorithm is completely independent of how the distances between the objects were computed. The algorithm starts with a *trivial clustering* $\mathcal{C}_0$ with n singleton clusters. At each iteration phase the algorithm searches those two clusters c_i and c_j that have the shortest 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. The merging of clusters is continued until there is only one cluster left. The output of the algorithm is the sequence of clusterings $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_{n-1}$.

The time and space complexity of Algorithm A.1 can be estimated as follows. The size of the distance matrix is $n \times n$ rows. This means that at each phase of the algorithm searching of the closest pair of clusters takes $O(n^2)$ time. Because there are n clustering phases, the time complexity of the whole algorithm is $O(n^3)$. On the other hand, because the distance matrix and the clusterings 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].

On Line 3 of Algorithm A.1 the pair of clusters that has the shortest distance is searched for. For finding such a pair of clusters we have to be able to define the distance between two clusters.

Definition A.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 the 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. If at least one of the clusters c_i and c_j consists of two or more

a)

b)

c)

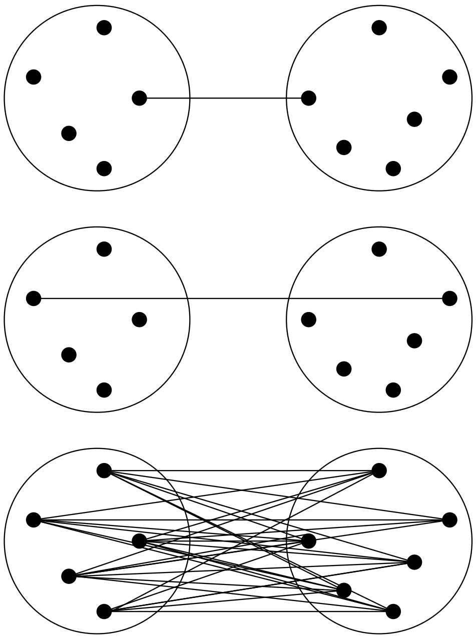


Figure A.2: Inter-cluster distances with different agglomerative methods: a) single linkage, b) complete linkage, and c) average linkage method.

objects, 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.,

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

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

The choice of the distance function d_{f} depends on the type of the objects considered. In defining distance between binary attributes, for example, any internal or external distance measure of attribute similarity would be used as d_{f}. The function F defining the inter-cluster distance between non-singleton clusters can also be chosen in different ways. In the following we describe briefly three agglomerative hierarchical clustering methods, namely the *single linkage*, *complete linkage*, and *average linkage* method that use different definitions for the inter-cluster distance between non-singleton clusters.

The single linkage method, also referred to 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.,

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

The method is called single linkage because in each clustering phase two clusters are merged by the single shortest link between them (Figure A.2a). This method produces clusters where every object in a cluster is more similar to some other object 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. 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 furthest members (Figure A.2b). That is, the inter-cluster distance between clusters c_i and c_j is

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

Now all the objects in a cluster are linked to each other at 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. In this method, forming elongated clusters is highly discouraged, and if the real clusters are elongated, the resulting clusters can be meaningless [DH73].

In the third agglomerative clustering method, the average linkage method, the inter-cluster distance of the clusters c_i and c_j is defined as

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

i.e., the distance is the mean of the pairwise distances between the objects in two clusters (Figure A.2c). This method is aimed at finding roughly ball-shaped clusters. In literature, this method is also referred to as *average linkage between merged clusters* [And73] and *unweighted pair-group average* method [DE83].

There are many reasons for the popularity of the hierarchical clustering methods. For the first, they are conceptually simple and their theoretical properties are well understood. Another 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 joined 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 that need to be examined at each clustering phase is reduced, and a computation time of the whole sequence of clusterings is rather short altogether. This property of keeping the merging and splitting decisions 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, it is impossible to correct it later.

Appendix B

Generating of synthetic event sequences

In our experiments on similarity between event types we used four synthetic event sequences. These sequences were generated as follows.

First, we produced a basic event sequence $\mathcal{S}$ where occurrence times of events were allowed to be between zero and a given parameter value n. We generated to this basic sequence $\mathcal{S}$ an event to a time $t \in \mathbb{N}$ with a probability of p, where $0 < p \ll 1$. In other words, in the basic sequence $\mathcal{S}$ there was no event at the time t with a probability of $1 - p$. This means that the probability distribution of the events generated followed the Bernoulli(p) distribution. Assuming that there should be an event at a time t, the type of the event was chosen to be one of the event types in a set $\{A_1, \ldots, A_q\}$ with a probability of $1/q$ where q was the number of the possible event types A_j. In turn, this means that the types of the events in the basic sequence $\mathcal{S}$ came from the discrete uniform distribution $DU(1, q)$.

After that, we added event showers to the basic sequence $\mathcal{S}$. We started this phase by generating r sets Z_i, i.e., sets $Z_1, Z_2, \ldots, Z_r$. Each of these sets Z_i contained k events. The types of these k events belonged to the set $\{A_1, \ldots, A_q\}$. Given the sets Z_i, we then generated a total of m event showers using Algorithm B.1. Each shower of events contained from one to k events, where the types of these events were chosen from a randomly chosen set Z_i. And at the end of the shower there was an event of either the type B or the type C with a probability of 0.5.

Finally, we generated to the sequences as many new events of a type D as there were already events of the type B. Occurrence times of these events of the type D were allowed to vary randomly between one and the given parameter value n.

Algorithm B.1 Event showers

Input: An event sequence $\mathcal{S}$, sets Z_i of k event types, and the parameter values m, n, r and u.
Output: An event sequence where m event showers have been added to the given sequence $\mathcal{S}$.
Method:

1. **for** $l = 1$ **to** m **do**
2. generate a random starting time t from the distribution $DU(1, n - k)$;
3. generate a random number i from the distribution $DU(1, r)$ for selecting a set Z_i;
4. **for** each $A_j \in Z_i$, $j = 1, \ldots, k$ **do**
5. add an event of the type A_j to the sequence $\mathcal{S}$ to the time t with a probability of $1 - u$;
6. t = t + 1;
7. **od**;
8. add with a probability of 0.5 either an event of a type B or C to the sequence $\mathcal{S}$ to the time t ;
9. **od**;

The parameter values used in generating the four synthetic event sequences used are given in Table B.1. In the sets of event types in these sequences there were always the event types B, C and D. However, the number q of event types A_j varied. Given the values of the parameter q in Table B.1, we can see that in the event sequence $\mathcal{S}_1$ the number of such event types was 10, in the sequence $\mathcal{S}_2$ 20, etc. This in turn means that in the first sequence there were a total of 13, in the second 23, in the third 33, and in the last sequence 43 event types. The total numbers of the events generated in each of the sequences were 6 038, 29 974, 94 913, and 142 433 events, respectively. Note that in each of the sequences the number of

Event sequence	n	p	q	r	k	m	u
$\mathcal{S}_1$	10 000	0.1	10	2	5	1000	0.3
$\mathcal{S}_2$	50 000	0.1	20	5	5	5000	0.3
$\mathcal{S}_3$	100 000	0.1	30	5	10	10 000	0.3
$\mathcal{S}_4$	150 000	0.1	40	5	10	15 000	0.3

Table B.1: The values of the different parameters used in generating the synthetic event sequences: the number n of time points, q of event types A_j, r of sets Z_i, k of event types in each set Z_i, and m of generated event showers, as well as the probability p of an event occurrence in the original sequence, and u of not adding an event of type A_j into an event shower.

events is less than the number of the possible time points; in the case of the event sequence $\mathcal{S}_2$ remarkably less than the number of time points n.