Une approche d'alignement à la problématique de la détection des activités habituelles

Rick Patrick Constantin Moritz

THESE

Présentée à :

L'Institut National des Sciences Appliquées de Rouen

En vue de l'obtention du grade de :

Docteur en Informatique

Par

Rick P. C. MORITZ

Routine Activity Extraction from Local Alignments in Mobile Phone Context Data

Soutenance le 05/02/2014

Devant le jury composé de :

Président	Christophe Garcia	Professeur des Universités	LIRIS, INSA Lyon
Rapporteurs	Bruno Crémilleux	Professeur des Universités	GREYC, Université de Caen
	Philippe Leray	Professeur des Universités	LINA, Université de Nantes
Directeur de Thèse	Michel Mainguenaud	Professeur des Universités	LITIS, INSA de Rouen
Co-Encadrants	Alexandre Pauchet	Maître de Conférences	LITIS, INSA de Rouen
	Grégoire Lefebvre	Ingénieur R&D, Docteur	Orange Labs Meylan

Normandie Université litis INSA ROUEN orange

Abstract

Humans are creatures of habit, often developing a routine for their day-to-day life. We propose a way to identify routine as regularities extracted from the context data of mobile phones. Mobile phones have become the *de facto* wearable sensor device, used and carried on person over most of the day. We study existing approaches to this problem, and deem past results not satisfactory, with regard to the ever richer context data available, that is not being exploited. Subsequently, we choose Lecroq et al.'s existing state of the art algorithm as basis for a set of modifications that render it suitable for the task, foremostly addressing performance issues. This algorithm is itself an evolution of a dynamic programming based local alignment algorithm, which is frequently used in biological sequence analysis. Our approach searches alignments in sequences of n-tuples of context data, which correspond to the user traces of routine activity. Our key enhancements to this algorithm are exploiting the sequential nature of the data to reduce algorithmic complexity by a factor equal to the number of data sources available, and using an early maximisation approach to reject non-optimal alignments in favour of optimal alignments.

We develop a generator of context-like data to allow us to evaluate our approach automatically, without relying on manual annotation of ground truth. Additionally, we collect and manually annotate a mobile phone context dataset to facilitate the evaluation of our algorithm. The results are promising, allowing us to prove the concept of our approach, while also outlining the limitations. Our contribution can be used as a filter for a domain expert (e.g. the user of the mobile phone himself) to determine whether a certain interval of data corresponds to an actual activity based habit, reducing the workload compared to a pure manual approach. The configurability of the algorithm allows the expert to have control over which patterns are filtered or retained.

Contents

Nomenclature . . . xii
1 Introduction . . . 1
2 State of the Art . . . 5
2.1 Context Datasets . . . 6
2.1.1 Reality Mining Dataset . . . 6
2.1.2 Lausanne Data Collection Campaign . . . 7
2.1.3 Discussion . . . 8
2.2 Machine Learning and Statistical Analysis . . . 9
2.2.1 Classification . . . 9
2.2.2 Clustering . . . 10
2.2.3 Probabilistic State Machines . . . 12
2.2.4 Discussion . . . 12
2.3 Sequence Mining . . . 13
2.3.1 Itemset Mining and Sequential Pattern Mining . . . 14
2.3.2 String Mining . . . 25
2.3.3 Comparison . . . 43
2.4 Conclusion and Direction . . . 46
3 Alignment of Sequences of n-Tuples . . . 47
3.1 Context Model . . . 47
3.1.1 Sequence of n-tuples . . . 47
3.1.2 Blocking and Sampling . . . 49
3.1.3 Meta-data . . . 51
3.1.4 Discussion . . . 54
3.2 Alignment Algorithm . . . 55

3.2.1 Contribution I: Reduction to n-tuple Problem . . . 57
3.2.2 Contribution II: Locally Optimal Alignments . . . 63
3.3 Discussion . . . 66
4 Experimental Validation and Results . . . 69
4.1 Synthetic Data Evaluation. . . . 69
4.1.1 Synthetic Data Generator and Dataset . . . 69
4.1.2 Synthetic Data Pattern Extraction Evaluation . . . 89
4.1.3 Analysis . . . 95
4.2 Real World Data Evaluation . . . 96
4.2.1 Data Collection Campaign . . . 96
4.2.2 Evaluation of Alignment Approach on Real Data . . . 102
4.2.3 Analysis . . . 106
5 Conclusions and Future Work . . . 109
Bibliography . . . 113
A Benchmarks of Supervised Classification Algorithms for Next Place Prediction . . 130
A.1 The Next Place Prediction Problem . . . 130
A.2 Dataset Analysis . . . 130
A.3 Next Visit Prediction. . . . 132
A.4 Conclusion . . . 136
B Details on the evaluation processes . . . 138
B.1 Real world data . . . 138
B.1.1 Similarity score tables . . . 138
B.1.2 Detailed results of the alignment algorithm evaluation on synthetic data . . . 141
B.1.3 Expert annotated patterns . . . 145

List of Figures

2.3.1 A WAP-tree and conditional WAP-trees for two subsequences c and ac (Source: Pei et al. [2000]) 20
2.3.2 Examples of different types of repeat-related features in a string. 26
2.3.3 Construction of a suffix tree from *abcbcabc*%. The numbers indicate both the order and the index of the current suffix being added. New leaves and nodes at each step are shaded. 28
2.3.4 Identification of longest repeats using Baker's algorithm in the complete suffix tree from Fig. 2.3.3. Steps 7, 8 and 9 of the tree construction are performed, adding the three shaded leaves. The order in which the leaves are added is given by the nodes and indicated by the superscripts of the vertex labels. 29
2.3.5 Original Needleman-Wunsch algorithm, as introduced by Needleman and Wunsch [1970]. The left table is filled with incrementation markers ("1") at each location where a pair of values match. In the right table, the values for all the cells have been determined, and the alignment highlighted by arrows and bold values. The alignment (arrows) starts at the maximum value and then follows the maximum values, upwards and to the left in the table. The "skips" between values 5 and 4, and 3 and 2 are indicative of deletions in the row-string. 33
2.3.6 Calculation of the local similarity using the dynamic programming approach. The preceding values in the table are either incremented by the substitution score, in the case of the diagonal predecessor, or decremented by the indel penalty. The maximum value is carried over for all successive operations. 34

2.3.7 Dynamic programming version of the Needleman-Wunsch algorithm. Non-matching substitutions have no penalty, matching substitutions score a value of 1 and insertions and deletions are penalised with a value of -1. The left table shows how a particular cell value is calculated from three predecessor values, the right table shows the complete table and highlights the series of maximal values which indicate the alignment operations. Note that the traceback follows substitutions over indels in cases of ambiguity (e.g. the 4→3 transition marked in the right table), as the value that was used to calculate the current value defines the path for the traceback, and not necessarily the maximum value. 36

2.3.8 Two accumulated similarity tables obtained using the Smith-Waterman algorithm. The left has been calculated using a similarity score of 1 for matches, and dissimilarity penalties of -2 for non-matching substitutions and indels. The right table has this penalty reduced to -1. In each case, the alignments with a similarity score of at least 3 have been highlighted. Note how the higher penalty leads to smaller, more local alignments. . . . 38

2.3.9 Row and column prefixes, and accompanying notation. 40

2.3.10 Three different ways of modelling context for different sequence mining approaches. "Data synchronisation" refers to the need to have a full set of sensor readings available at every time step, irrespective of different sampling frequencies of the actual sensors. 42

3.1.1 The transformation of continuous context $C(t)$ into context data in a sequential model. 48

3.1.2 Linking w- and z-axes into a single dimension, due to translational symmetry (i.e.: every set of discrete steps taken along the z-axis is replicated identically in the w-axis). 49

3.1.3 Splitting a long sequence of context data into blocks of roughly equal (but not necessarily equal) lengths. 50

3.1.4 The reduction of the size of individual accumulated similarity score tables which can be obtained by blocking. a and b are two sequences, with the local similarity scores contained in the volume spanned between them. . . 51

3.1.5 A sample substitution similarity score table from the set of n tables. . . . 52

3.1.6 Two intervals of identical values but different lengths can be aligned by assigning positive similarity score to insertions and deletions of identical values. 53

3.2.1 Two example context sequences, the resulting context data sequences and an alignment (cells underlaid in grey) that corresponds to two similar subsequences. 55

3.2.2 Alignment of two context sequences a and b from a corpus. 56

3.2.3 Selection of tuple elements in a sequence using reverse addressing. The origin of the coordinate system used here is in the lower right end of the sequence. From there the first coordinate is incremented when moving one element to left, and the second when moving one element upwards. . 58

3.2.4 Example of the $\bullet_{i,,j}$ operator. If the operator is used on the same sequence of n-tuples, with one pair of indices being higher than the other, the smaller result is a subset of the larger one. In this case the result underlaid in black is a subset of the result underlaid in grey. 59

3.2.5 Selection of a column and row using the $\uparrow$ and $\leftarrow$ operators. 60

3.2.6 Calculation of a column similarity score. Two columns are extracted from two sequences, then aligned in inverse order, up to the point where the similarity score reaches zero. The maximum similarity from this interval is the similarity value we assign these two columns. 61

3.2.7 The local dependencies of the calculation of a local similarity value. To the left in the three-dimensional table of accumulated similarity scores T and to the right the equivalent representation in the domain of the two sequences a and b. Operations V and VI – insertion and deletion – are one-dimensional operations and therefore appear only either in a or on b. 63

3.2.8 The rightmost (n-th) z-slice of a table T. The axes of the plane correspond to the temporal axes of the sequences. The height and colour are representations of the accumulated similarity value in the cell in T. The white profile line denotes the plane of an arbitrarily chosen MASS. The highest value is denoted by an X. 64

3.2.9 The local neighbourhood of a node (in the centre of the shown cube) in the accumulated score table T. The shown cube consists of 27 sub cubes, the one at the centre being the "home node". The other 26 surrounding it, are the direct neighbours. A partial spatial subdivision is shown in the lower left corner to illustrate the notion of "radius". 66

3.2.10 Accumulated similarity score table for the alignment operations of the context data corresponding to two consecutive days. Each graph represents one of the five slices ($n = 5$) of the z-axis, starting at one at the top, incrementing to five at the bottom right. Height and colour correspond to local similarity values. The x - y plane is spanned by the temporal axes of the sequences. Note multiple peaks at different heights. 67

4.1.1 Stream, pattern and cell for a simulated 4-tuple dataset. The stream consists of patterns (coloured) and random data (light grey). Each pattern consists of defined values (blue) and undefined values (white). 71

4.1.2 Four alphabets $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4$ with respective spectra 5, 6, 4 and 2. N.B.: Different symbols across different sensors (e.g. "A", "1", "a", "0") are used here for illustration purposes only, symbols are actually represented by their indices (i.e. as in $\mathcal{A}_2$). 72

4.1.3 A set of patterns, $\mathfrak{P} = \{X_1, \ldots, X_m\}$, with n data sources and lengths $l_1, \ldots, l_m$, and a pattern element $x = X_{m,n,1} \in \mathcal{A}_n$. 74

4.1.4 Illustration of dependence coefficient determination during creation of similar patterns. The right vectors are created from the values of the left vectors, through linear interpolation and scaling. 79

4.1.5 Noise is applied to symbol x from alphabet $\mathcal{A}_1$ using transition cost matrix T_1 and transition cost vectors $\vec{b_1}, \vec{d_1}, \vec{e_1}$. The resulting value is $y = 4$. This example uses integer values for easier reading and comprehension, in general the random values are real values. The process starts with value $x_{2,1,1} = 2$ from pattern X_2, extracts the corresponding row from T_1 and selects the smallest absolute value in the sum of the extracted row and a random vector, to determine the index of the element to replace x. 84

4.1.6 Extract of a possible resulting stream with $n = 4$ and $m = 3$ different patterns (blue, yellow and green) and noise (red). I and D indicate cells in which insertions or deletions have happened. y is the symbol that has been generated in Fig. 4.1.5. 85

4.1.7 Evaluation scoring example. 90

4.1.8 The ratios of the number of alignments to the combinatorially expected number of identical pairs of patterns in two different files, for all five datasets and all 27 experiments. As the number of patterns rises, the expected number of pairs gets lower, and the rate of alignments gets closer to the number of pairs.
*Note: The values for dataset 3 are on another scale than the others. . . . 93
4.2.1 Clustering of locations into places of the data gathered by one subject over a one week timespan. Artificial jitter has been introduced to the data, to better visualise data density. Note how nine incidences in the top left are clustered together with the other points towards the top left. The low number of incidences renders this error negligible. 100
4.2.2 Visualisation of an extract of the captured data, with similar intervals marked. This extract covers the morning period of two consecutive days. The y-axis represents the different discretised context values. The values shown in this extract are light intensity on a logarithmic scale (loglux), the state of the proximity sensor (prox), the location cluster id (place), the battery level (batt) and the orientation of the phone (ori). 101
4.2.3 Alignment of three pre-selected patterns with campaign data in 58 slices corresponding to 24-hour periods of 5-tuples. 103
4.2.4 Mean precision and recall for the instances in the corpus ("B" - part of the alignment) for the 24 test configurations. From left to right are the different values of MASS used. With rising MASS, both precision and recall increase. The rightmost bars are fewer, as no alignments were made with MASS values 1,700 and above for pattern 3, and no alignments were made with MASS values 3,000 for pattern 1. Due to the smaller number of elements contained in pattern 3, it has higher values than patterns 1 and 2 for the same MASS. 107

A.3.1 Results for the Naive Bayes Classification Algorithm 134
A.3.2 Results for the LogitBoost Adaptive Decision Tree Classification Algorithm 135
A.3.3 Results for the SMO Classification Algorithm without Feature Selection Filters 135
A.3.4 Results for the SMO Classification Algorithm with Feature Selection Filters 136

B.1.1 Reference pattern 1 - evening and night at the week-end home. 146
B.1.2 Reference pattern 2 - morning and transit to work on a week-day. 147
B.1.3 Reference pattern 3 - at work, telephone stored or turned over. 148

List of Tables

2.1 Comparison of SPM and string mining 45

4.1 Configuration of the five datasets. 87
4.2 Configuration values for the 27 experiments making up a dataset. 88
4.3 Transition matrix T used for the generation of the test datasets. 88
4.4 Key results: averages and standard deviations across all configurations. . 91
4.5 Average Pattern Size for Dataset One 92
4.6 Key results: averages and standard deviations across all configurations for the state of the art algorithm 95
4.7 Evaluation results. A is the search pattern, B is the instance in the corpus. 105

A.1 Classifier and Filter Configurations Tested 133

B.1 Similarity score table for luminance 139
B.2 Similarity score table for proximity 139
B.3 Similarity score table for places 139
B.4 Similarity score table for battery levels 140
B.5 Similarity score table for orientation values 140

Nomenclature

API	Application Programming Interface. A set of exposed functions available to programmers to make use of an established software platform.
BIDE	BI-Directional Extension based frequent closed sequence mining. A closed sequential pattern mining algorithm by Wang and Han (2004)
BLAST	Basic Local Alignment Search Tool. An approximative but fast local alignment algorithm by Altschul et al. (1990).
BLOSUM	BLOcks SUbstitution Matrix - A substitution scoring matrix for the alignment of proteins. The matrix entries are determined by statistical analysis of a database of evolutionary observations.
BT	Backtrack - Second step of a dynamic programming algorithm; in our case a trace along elements in a score table, marking the operations required to convert one sequence into another.
DNA	Desoxyribonucleic acid. A molecule that encodes genetic information in a sequence.
FASTA	fast - All. A sequence mining software suite developed by Lipman and Pearson (1985).
FP-growth	Frequent Pattern; Introduced with the FP-growth algorithm by Han et al. (2000).
GPS	Global Positioning System – Satellite based time-of-flight positioning system

GSM	Global System for Mobile communication – European standard for second generation digital cellular networks for mobile phones.
GSP	Generalized Sequential Patterns. An Apriori-type sequential pattern mining algorithm developed by Agrawal and Srikant (1996)
indel	Insertions and deletions in sequences. A single term is sufficient to describe both these inverse operations, due to the symmetry of the alignment operation with respect to the order of the input sequences.
LAN	Local Area Network - a computer network with scope limited to a small geographic entity and using local addressing.
LZ	A set of compression algorithms proposed by Ziv and Lempel in 1977 and 1978.
MASS	Minimum Accumulated Similarity Score. This value is the criterium whether two sequences are to be considered similar or not.
MDC	Nokia Mobile Data Challenge. A data mining and visualisation competition held over the first half of 2012, on a mobile phone dataset gathered by Nokia. The dataset is explained in Subsection 2.1.2, and one of the competition tasks is presented in Annex A.
RMD	Reality Mining Dataset. See subsection 2.1.1.
RNA	Ribonucleic acid. A functional molecule that encodes genetic information in a sequence.
sensor	In this work: a source of discrete context data. This can be anything from a Boolean indicator to a fusion of multiple semantically linked context data readings, such as orientation, linking the data of an accelerometer and magnetometer.
SPADE	Sequential PAttern Discovery using Equivalence classes. A sequential pattern mining algorithm proposed by Zaki (1998)
SPAM	Sequential PAttern Mining. An algorithm proposed by Ayres et al. (2002) for finding all frequent sequences within a transactional database.
SPM	Sequential Pattern Mining

WAP	Web Access Pattern. A user's trace when navigating across websites.
WLAN	Wireless Local Area Network. A term encompassing networking as defined in IEEE standards group 802.11.

Introduction

Multiple studies have shown that humans are creatures of habit – from Heidegger's musings in "Sein und Zeit" to the more concrete works of Gonzalez et al. [2008] and Eagle and Pentland [2009]. Although routine is not universal, for those that adhere to it, it provides a stable framework of activities. If a computer system can be made to be aware of this routine, then it can be proactive and anticipatory in nature, preparing for commonly encountered situations, or even alerting the user when routine is unexpectedly broken at some point. Routine can also be used to enhance a computing system which adapts itself to current events. When historical data can be linked to the present state, activities and contexts can be recognised with increased certainty. All this is beneficial, when developing ubiquitous computing systems – specifically with regard to ambient, passive interfaces.

We assume that habits (i.e. activity patterns indicating routine) can be derived from user context data. "Context" classically refers to the entirety of the environment of a human-computer interaction – slightly varying definitions have been proposed by Schmidt et al. [1999], Chalmers [2004] and Dourish [2004]. Examples of context data are factors such as physical environment (e.g. place, light, noise), social environment, system status and simultaneous tasks. For the purpose of determining routine, we are primarily interested in specific aspects of context, that are characteristic to an activity. This does not necessarily imply that the context data allows one to *infer* the actual activity, but it is sufficient to identify recurrences. One theme of this thesis is in fact that of the title of Schmidt et al. [1999] "There is more to context than location": there exists a large body of work on spatio-temporal routine detection, but our aim is to go further, and include other context data to obtain a model of routine that is richer in information and higher in resolution.

Currently, the most widespread device in use with the capability to observe a reasonable amount of context, is the mobile phone. These phones are equipped with a number of sensors and an almost constant network connectivity which allows queries to nearby fixed sensors. For many people, their mobile phone is a constant companion allowing the capture of context data for a large number of activities. Furthermore, as multi-purpose mobile computing terminals, these same phones also form the interface to the computing system that benefits the most from having access to routine information. With regard to the definitions of context previously given, there is one further remark to make: context is defined as being linked to an interaction. At first glance, capturing data over the course of a day, to observe routine, is a passive process and does not fall under the traditional view of an interaction. But – within the concept of ambient interfaces – even just carrying a mobile phone (or in fact, even *not* carrying it) becomes an interaction, *because* we instrument the phone. Therefore, this implicit interaction happens within a context, which is duly recorded.

Routine conveys important information all by itself, but can also be used to enhance current context information. A key use of context data is directly on mobile devices, which are situated within a certain context, used within a certain context and thus ideally exploit this data to better fulfil their purpose, as well as adapt and respond to changes in context. One pertinent example for such a mobile application that uses routine and current context data, is the Magitti leisure guide [Bellotti et al., 2008]. Similar applications in the domain of prediction and recommendation are presented by Anand and Mobasher [2007]. Wesson et al. [2010] discuss the benefits of adapting the mobile user interface of a computing system to the current context. Baldauf et al. [2007]'s survey of context aware systems essentially covers many more examples, where having additional context information based on a routine model can improve the user experience. The spectrum of approaches covers safety, security, entertainment and education. Mobile guidance and navigation systems can also be made to be context sensitive and routine dependent [Brush et al., 2010, Li and Willis, 2006].

Our technical goal is to detect patterns in the context data of a mobile phone user, with the expectation that patterns in the data correspond to routine activities. Raw context data can be understood to be a number of time series of physical values. Based on this representation, there exist multiple ways to analyse the data: Statistical approaches based on state transition probabilities (e.g. Ashbrook and Starner [2002]), database oriented approaches based on sequential apparition frequency (e.g. Giannotti et al. [2007]) and sequence based approaches based on edit distances and similarity measures (e.g. Sigg et al. [2010]). As the title of the work suggests, we ultimately pursue an

approach that performs approximate string matching by aligning pairs of sequences. This algorithm determines local similarity scores for sequences of context data – we suppose that highly similar sequences of context data correspond to similar contexts and thus to actions of the same type. But we must also consider potential shortcomings of the data on which we base our study: sensor noise may render an activity difficult to recognise, and an activity is not repeated identically every time, leading to potential variations in duration and values in the corresponding context data, which must also be accounted for.

Our alignment algorithm – an adapted version of the algorithm introduced by Lecroq et al. [2012] – is based on a representation of context data as a sequence of n-tuples. In this structure we calculate local alignments. To determine if these alignments correspond to context patterns, we devise an evaluation procedure. This procedure is in one case applied to synthetic data, generated according to predefined parameters, and in another to real world data which we gathered during a data collection campaign. The results from a large spectrum of test conditions allows us to characterise the suitability of our algorithm for the routine activity detection problem. The final results indicate that the algorithm reliably performs as a filter for context data, to limit the function of a domain expert to approving or rejecting the extracted data as being correspondent to a routine activity. Conversely, there are limitations to using it as a general, fully automated routine extraction approach.

This work is structured as follows:

- In the second chapter, we examine the state of the art for both available context datasets and pattern detection approaches. We study the datasets for their suitability for routine context extraction and their data collection methods. The pattern detection approaches are divided into statistical and sequence mining approaches. We briefly discuss statistical approaches, and explore the various sequence mining approaches in more detail. We split the latter field into sequential pattern mining approaches, which take a database oriented view on sequences, and on string mining, which is focused on comparing long, individual sequences. We conclude this chapter with a direct comparison of the most promising approaches and an argumentation for our choice of approach, as well as identifying the limitations upon which our solution has to improve.

- The third chapter introduces our context model and algorithmic contributions. The context model defines a similarity measure for a pair of context sequences, which the alignment algorithm then uses to find maximally similar subsequences

from a pair of sequences. Our algorithmic contribution is divided into two parts. The first part reduces algorithmic complexity by simplifying the state of the art approach, the second part details our local maximisation approach to further increase performance.

- The fourth chapter is dedicated to the experimental evaluation of our approach, on synthetic and real world data. It describes a data generator we developed with the goal of creating well-understood data with context-data-like characteristics and a high degree of configurability to allow the reproduction of a variety of scenarios. We also document our results obtained on five different such scenarios. A set of variable parameters for each scenario allows us to gain a further insight on the theoretical performance of our algorithm and its behaviour under varied conditions. We also present our data collection campaign in some detail. We used part of that data for a second evaluation, which is also described in this chapter.

- The final section gives a summary of our contributions and results, presents our conclusions and discusses limitations to our approach as well as potential future work to address these limitations and propose extensions.

State of the Art

As we laid out in the introduction, our approach to identifying routine activities is to find patterns in time series of context data. In this section we present and discuss different possible approaches, which should theoretically reveal such temporal patterns. We briefly revisit the notion of "context" and examine how context data is represented in actual context datasets. By class of approach, we present existing routine activity detection solutions.

We examine the varying approaches for a number of criteria, relevant to our task. These criteria are:

1. The ability to process data that has the volumetric characteristics of context data, regarding both algorithmic complexity and real world constraints;
2. The degree of pre-treatment of context data required;
3. The flexibility of the underlying model, to accept situation-specific tuning;
4. The resilience of the approach against the inherent noise and unreliability in sensors and slight variations in how activities are repeated;
5. The impact of temporal effects, such as desynchronisation of data and dilation or contraction of the time-scale of an activity;
6. The effort required to parametrize the underlying model and
7. The suitability to a (partial) deployment on mobile platforms.

We will refer to these criteria by their number in the following sections, whenever a relevant bit of information is touched upon.

Our examination of the state of the art is split into three sections. First, we take a look at existing context datasets, to situate our goal with regard to available data. Then, we survey the state of the art for relevant methods that allow us to match, compare or extract patterns in time series data, by studying two fields of specific interest: Statistical approaches to pattern mining and sequence mining based approaches. Finally, we cross-compare the most suitable approaches and datasets, to determine the course for this work.

2.1 Context Datasets

When examining user behaviour over the long term, a large dataset of context data is crucial to verify our hypothesis. In this section, we look at a number of different context datasets to learn more about *available* context data, and the utility of this data for our task.

A small dataset, that can be understood to be a pioneering precursor, is that of Raento [2004]. Another early dataset by Mäntyjärvi et al. [2004] was limited to a strict set of scenarios, and did not actually consist of a recording of real world data. Kotz and Essien [2005] present a simple location-focused dataset, obtained by studying the movement of wireless network users of Dartmouth college.

Two specific context data sets have caught our attention, the reality mining dataset (RMD [Eagle and Pentland, 2005]), and the Nokia mobile data challenge (MDC [Kiukkonen et al., 2010]) dataset. They represent, respectively, the first large scale mobile phone dataset ever published as well as the largest and most complete dataset currently available.

2.1.1 Reality Mining Dataset

The RMD is the first large scale mobile phone context data set that has been collected. Due to its public availability, it stands as a reference dataset when it comes to the detection of routine user behaviour. It was designed with this goal in mind and used in the work of Eagle and Pentland [2009] to determine behavioural routines.

The dataset is the result of a broad (100 participants) mobile phone data acquisition campaign the data from which was made available to the scientific community. The RMD was established using the Context Phone application developed by Raento et al. [2005]. It contains primarily finely grained location information, but also application and communication use logs and relative proximity indicators among study subjects.

The specific protocol was established as follows: One hundred MIT students and members of faculty staff were given Nokia smart phones, pre-installed with several applications that track context. This context data comprises call meta data, the set of Bluetooth devices in range, the ID of the connected cellular radio tower, application usage and some phone status variables, such as whether the phone is idling or charging.

This data was collected over an academic year, aggregating 450,000 hours of data, with Bluetooth data being logged in five minute intervals. The researchers report some data loss (around 15%) due to some subjects deactivating their phones during the night, and also due to some data corruption events. Finally, there are incidents of forgotten phones, where the phone records data, which has no association with the actual activities of the test subject.

In addition to the context data, there is data from surveys of subsets of the test group to establish some meta-data (relationships between study participants and social activities) to the main data gathered by the phones. Some subjects also kept more detailed activity diaries over part of the data gathering period.

2.1.2 Lausanne Data Collection Campaign

This dataset [Kiukkonen et al., 2010] includes data from 200 users from the Lake Geneva region in Switzerland and consists of the data from around 80 different smart phone context data sources, recorded over a period of over one year. This data was made available to researchers within the framework of the Mobile Data Challenge [Laurila et al., 2012], but is not readily available to the public.

The gathered data can be divided into four groups: Location data, media-related data, social interaction data and behavioural data. The central physical data present in the set are 3D accelerometer readings, location and sound measurements. Additionally logs of applications, communications and the operating system are kept, alongside the agenda and contact list.

A key distinction of this dataset is the heterogeneous study group. Where the RMD only studied students and faculty staff members from the Media Lab, as well as students of one other lab, this campaign stretches much wider. By consequence, the social graph among the participants is much less dense.

Another difference lies in the precision of the location data. Cell tower data is notoriously imprecise, discriminating at around the city block level, whereas more powerful modern phones are equipped with GPS which can reduce this to roughly the street address level, even without further augmentation with wireless networks. Using WLAN access points – as done for this study – increases this precision further, especially in

urban zones. In comparison, the Bluetooth augmentation chosen for the RMD only works in select installations outfitted with a beacon.

This much higher fidelity of the data leads to a much larger challenge, when distributing this data, as pure anonymisation no longer suffices to protect the identity of the subjects.

Furthermore, the higher potential fidelity also implies a potential higher use of the battery. Non-intrusiveness in the routine usage of the phone is a key requirement, as otherwise the impact of adapting to the measuring device would create behaviours and data not corresponding to the reality of the test subject's day to day life. In an attempt to mitigate these effects, each data source has its own update frequency, in an adaptive event-based system. This allows the sampling to run throughout the day.

To minimise data losses, collected data is transmitted to a database server via wireless networks twice a day.

With regard to ground truth, an initial questionnaire establishes some demographic and social information, but no further information is acquired.

2.1.3 Discussion

A key limitation to the present datasets, is that none of the bases contains any ground truth regarding context routine. Most do not include any manual annotation with semantic data at all. In fact, the large scale nature is somewhat at odds with such demands, as the effort to manually annotate a large dataset by someone other than the subject is immense. Conversely, asking a subject to annotate his or her own data introduces a higher level of effort required to take part in such a campaign and decreases the number of participants.

Another problem is that most datasets are limited to location data, and lack many of the physical sensors present in more modern phones, that we presume to capture physical context more accurately. Although the Nokia MDC-dataset does provide accelerometric data and some sound samples, modern smart phones are also equipped with gyroscopes, compasses, light meters and proximity sensors. Conversely, the approach taken by Nokia for the Lausanne data collection campaign includes a vast number of sensors, some of which are not necessarily of interest for the routine activity detection task, increasing the complexity of the required pre-treatment (2). The asynchronous nature of data collection preserves relative temporal structures in high detail (5).

Finally, the acquisition platforms used for these two previous datasets have been obsoleted by the development of the mobile phone market. Therefore, if a new dataset would be a requirement, this would entail a new development of the data collection software

suite.

The creation of a new mobile context dataset that contains at least sufficient physical sensor values therefore appears possible and necessary. Many of the paradigms put forward for the different collection campaigns remain valid, such as the principle of non-intrusiveness and the need to assure data integrity. Alternatively, it is common to use synthetic data to verify the behaviour of an algorithm. For this type of data, ground truth of the generation step can be used to address the lack of ground truth in real context datasets.

Once a solution to gather the required physical context data is achieved, the next challenge is to find routine data within it.

2.2 Machine Learning and Statistical Analysis

Classic statistical machine learning approaches are a mainstay of much of the work of context data analysis. We focus on three key categories: classification as a means of context and activity recognition and prediction, clustering to determine similarities in data and probabilistic state machines – specifically (hidden) Markov models – as a means of modelling routine.

2.2.1 Classification

Classification algorithms are a class of algorithms that use a set of examples to learn rules that are then applied to determine which label is to be associated to a measurement. Typically, this is represented by the determination of a characteristic vector over the feature space of the observed events.

We discuss two different usages of these types of algorithm, within the scope of the routine activity detection problem: in activity recognition, a current set of measurements of context data are used to determine the current activity; in activity or place prediction, the current state of system is used as a characteristic vector for predicting a future state.

2.2.1.1 Activity Recognition

Supervised classification is the principal component in most activity recognition systems. They function by assigning a characteristic vector to each element of a set of activities, and then estimating from actual measurements the current activity.

In the field of mobile device based activity recognition, supervised classification has been used (for example) in the works of Choudhury et al. [2008], Lester et al. [2006],

Berchtold et al. [2010] and Sigg et al. [2010]. Activity recognition by itself does not solve the routine activity detection problem, but a well functioning activity detector can reduce the complexity of the problem by orders of magnitude by transforming it from a multi-dimensional problem into a uni-dimensional one (1,2). We can nonetheless state a limitation to this approach, in that a supervised activity detection requires knowledge of each of the activities that are to be discerned (6). This renders this approach of limited use, when trying to solve a general version of the routine activity detection problem. Furthermore, feature vectors are understood as a unit, therefore algorithms based on them cannot work around desynchronized data (4,5).

2.2.1.2 Activity and Place Prediction

Another potential application of supervised classification is in the field of predicting *future* activities. The approach is similar to activity detection, with the difference that characteristic vectors are assigned to subsequent activities or places. This type of approach was frequently chosen by entries for the MDC "next location prediction" task [Etter et al., 2012, Wang and Prabhala, 2012, Gao et al., 2012], as for that task the known data was restricted to the context data obtained during the "visit" to the directly preceding place. The most successful approaches relies on heuristics which reject certain subsets of the data by declaring it out-of-date information with regard to the prediction task. The benchmarks of three state of the art algorithms – naive Bayes [John and Langley, 1995], alternating decision trees [Holmes et al., 2002] and a sequential minimal optimization based approach [Platt, 1999] – on the MDC prediction task are available in Annex A.

The Magitti [Bellotti et al., 2008] recommendation system predicts next activities using a supervised classification approach [Partridge and Price, 2009]. The system is based on the assignment of a leisure activity (e.g. "eat" or "watch") to corresponding locations in town instead of having a location specific activity model. This meta-activity approach avoids the problem of being limited to recognising specific activities, but consequently the result is too general for many applications outside the activity adviser use case they target (3).

2.2.2 Clustering

Where classification assigns one of a pre-defined set of labels to a sample of data, clustering attempts to group a set of data points via a pre-defined distance metric. There are three aspects to clustering that are of relevance to the routine activity detection prob-

lem: Using clustering to segment sensor data, using clustering to segment sequences and clustering entire sequences. The first is an important step towards obtaining a discrete context model, the second helps identifying atomic activities, and the third is a routine activity extraction approach.

2.2.2.1 Context Data Clustering

Context data clustering is a useful first pre-treatment step to reduce the complexity of the routine activity detection problem (1), by transforming the problem from one on continuous data (and essentially a signal processing problem) into a problem over discrete data (2,4). This has been used by Ashbrook and Starner [2002] to derive semantically relevant places from a user's spatio-temporal positioning data, which then allows the construction of a state-based predictor on the transformed data. Their choice of clustering algorithm was the k-means clustering algorithm. In their article they also meet one of the limitations inherent to clustering algorithms: the results often have to be manually verified, or selected from a number of different configurations (3,6).

2.2.2.2 Sequence Segmentation

Sequence clustering can be performed to segment long sequences of context data into locally self-similar subsequences [Clarkson and Pentland, 1999, Himberg et al., 2001]. This can be used as a semi-automatic pre-treatment step to address (1,2,6,7) in routine activity detection approaches, at some potential disadvantage to (4,5). In general, sequence clustering with a simple Euclidean distance does not produce meaningful results [Keogh et al., 2003], requiring the choice of a meaningful metric, before implementation.

2.2.2.3 Sequence Clustering

Clustering can also be used to group subsequences by inter-similarity and thus identify patterns. This has been shown by Laasonen [2005] to be a potential avenue of establishing a routine based user model and subsequently recognising and predicting a mobile user's routes. This works by classifying some of the locations visited by a user as *bases*, which are locations where a user rests, and others as transient points. In this case, a place corresponds to the currently connected network cell. An edit distance and item similarity measure serves as basis for the clustering model. Katsaros et al. [2003] have shown a similar sequence clustering solution to predict locations, but without the distinction of location classes. The distance measure for the hierarchical clustering which they employ, is a weighted edit distance. This can have good performance with regard to (4,5) at some

cost to (6). As a statistical approach, there is also a limitation with regard to (1), in that a certain minimal amount of data is required to obtain clusters with acceptable confidence values.

2.2.3 Probabilistic State Machines

In the third type of approach, we look at different kinds of probabilistic state machines, which are relevant for user activity recognition and prediction.

A relevant example of using hidden Markov models [Baum et al., 1970] to find routine in context is the work of Clarkson [2003], who studied video streams taken from two worn cameras, with the goal of identifying context recurrences. They use an alignment based approach, similar to dynamic time warping [Myers and Rabiner, 1981]. Ashbrook and Starner [2002] use second order Markov models to predict next locations. This required clustering of geolocation data into discrete locations, which where then studied for transition probabilities.

In the work of Song et al. [2004], multiple predictors based on probabilistic state machines (Markov models and LZ compression [Ziv and Lempel, 1978]) are evaluated on a common dataset from the Dartmouth campus wireless network [Kotz and Essien, 2005]. A state machine approach requires a sequence segmentation to be viable (6). This means that activities are identified either based on supervised activity recognition or based on sequence segmentation. A common drawback to all approaches outlined in this subsection, is that they are not truly suited to treating multiple data sources in parallel (1,5).

2.2.4 Discussion

Within the frame of the routine activity detection problem, statistical approaches can be divided into two categories of application: on the one hand there are data pre-treatment approaches, that reduce the complexity of the actual routine extraction process, and on the other hand there are statistical models that encode routine activity.

The former can be used at different levels of granularity, and comprise classification, clustering and segmentation. A high abstraction approach – as often chosen in the literature we presented – can simplify the routine activity detection problem to the point where it is merely the extraction of frequent state transitions. While in theory this is advantageous with regard to (1), (6) and (7), it invariably has drawbacks, when applying criteria (2), (3) and (5). A low abstraction level has the inverse consequences. Therefore, the choice of abstraction level has to be carefully weighed, when developing

a discrete model.

The routine activity detection approaches – using sequence clustering or probabilistic state machines – are relatively simple propositions (1) which depend heavily on accurate preprocessing (2). The central weakness of these approaches – when used on rich context data – is that they use a simple model derived from data that has been heavily preprocessed and abstracted, which results in abstract routine models. These are easy to interpret but hard to validate against the actual data.

Furthermore, there is a whole host of limitations to automatic context data abstraction that essentially make this class of approaches not suitable for context data that has multiple independent data sources. When working at a finely grained level, with less abstraction, the models become harder to interpret, and lose their appealing simplicity. This downside leads us to examine approaches that are more suited to this kind of data representation: sequence mining algorithms.

2.3 Sequence Mining

Context data is inherently of sequential nature: for each aspect of context, one state follows another and each activity is followed by another. Sequence mining is a subset of data mining, which consists of approaches specifically targeted at sequential data.

The field of sequence mining offers a variety of algorithms designed to discover all kinds of features in sequences of discrete elements. As laid out in the introduction, our primary interest is pattern detection. There are two schools of thought, with regard to this problem:

1. Sequential pattern mining (SPM) takes an itemset mining and association rule learning approach to finding such patterns, by looking at the frequency of occurrence of subsequences in a corpus.

2. String mining is an approach that has its roots in bioinformatics, and can be seen as a generalisation of Hamming and Levenshtein distances. We are particularly interested in alignment problems, where the goal is the identification of similar sequences.

In the following two sections, we will present different approaches and applications of sequential pattern mining and string mining. In the third subsection we will compare the particular advantages and challenges of each class of approach, and evaluate them against our set of criteria.

2.3.1 Itemset Mining and Sequential Pattern Mining

The aim of sequential pattern mining (SPM) is to find within a database of sequences of itemsets (i.e. sets of discrete elements) a subset of interesting sequences. Within the context of the routine activity detection problem, this criterion of interest is that the sequences appear frequently. Sequences in this case are not limited to consecutive elements, but are based on the notion "event Y occurs after event X, within an interval window δ". There are several ways to formulate the routine activity detection problem in terms of an SPM problem, each suited to a specific group of algorithms from the field.

First, context could be abstracted to fit the "sequence of itemsets" paradigm. There is a wealth of classic approaches that find potentially non-consecutive exact sequential patterns. A first challenge in adapting the problem to this formulation lies in finding a useful time-discretisation and temporal splitting interval. The former is achieved by having each sensor value discretised into an item, and by grouping measurements during a predefined time interval into itemsets. The latter requires study of how gravely non-consecutive patterns impact the results. By shortening the sequences, such patterns become less of an issue, but other, larger patterns may disappear. Additionally, there is the question of which level of support is desired as basis for the result.

An early review of examination of patterns in sequential data was performed by Laird [1993]. At this stage, the focus of research was mainly on prediction, extrapolation and modelling of time series.

In the same year, itemset mining was introduced by Agrawal et al. [1993]. This forms the basis for all of the following SPM approaches, which is why we take a closer look at the key approaches of the field. The following classification of the approaches is based on the taxonomies proposed by Mabroukeh and Ezeife [2010] and Mooney and Roddick [2013]. We assign the following classes to differentiate between approaches:

- Apriori-based (including variants for closed patterns or hierarchical data),
- Pattern-growth (including variants for closed patterns),
- Vertical data structures,
- Early-Pruning and
- Hybrid;

and then we also discuss variations of these which take into account hierarchical metadata, specifically and exclusively mine closed or maximal frequent patterns, or mine multidi-

mensional or approximate patterns. All algorithms and challenges identified in the field of itemset mining can be transferred almost exactly to sequential pattern mining.

2.3.1.1 Itemset Mining

Itemset mining has as prerequisite the presence of a database which is partitioned according to a primary criterion (e.g. a user ID or a calendar day). The goal is to find subsets of data that occur multiple times among the sets of data points connected to each instance of the criterion. An example relevant to the field of mobile and context sensitive computing is a database consisting of entries that have been generated by logging discretised context data of a user over time. Splitting the database to obtain individual data per hour, and then searching for co-occurrences of certain context values in these sets across different hours, shall determine which context states are frequently encountered together in temporal proximity.

Once the frequency of appearance of such an item combination is above a certain threshold, the set is considered to be significant. Furthermore, there is an interest in maximising these sets: a single element that appears across multiple sets is of little interest, whereas large sets carry more information. Once these sets are identified, association rules can be created among them.

Association rules can be understood as a way to encode the conditional probability of the presence of an element in an itemset, given a set of elements already present in that set. For example, let there be three context states a, b and c(e.g. loud noise, bright light and medium movement speed) that were recorded during one hour. Furthermore, let the set (a, b, c) appear frequently (and maximally) in a database. Then, it is possible to predict that if a and c are present in one set, that b might also be present in this set (e.g. because all three are present when the user rides his motorcycle on a sunny day) based on the *support* of the set (a, b, c) (i.e. how often the elements appear together, relative to the overall number of sets created by splitting the database along the key criterion). In terms of context prediction, this means that it is possible to predict context event b as a function of the presence of context events a and c. Clearly, this is limited in scope with regard to the routine activity detection problem, but the close link to sequential pattern mining makes it deserving of a study, to introduce the concepts used further on.

Apriori-Based Algorithms The itemset mining problem has first been approached by means of the Apriori-class of algorithms [Agrawal and Srikant, 1994, Mannila et al., 1994]. The key property on which the Apriori-class algorithms are built is the downward closure: the fact that each frequent itemset consists of sub-itemsets that are all frequent,

and inversely, extending a non-frequent itemset by any item results in another infrequent itemset. This can be exploited to first find frequent elements, and then iteratively extend these candidates to sets, element by element, until the support frequency-threshold is no longer attained. Han et al. [2007] offer an extensive list of further improvements to this approach, with regard to distributed and parallel approaches, as well as general efficiency improvements. The key limitation to this approach lies in the high number of database scans – one for each growth operation – that are necessary to determine the support value for each of the valid candidates that have been generated. Compared to the naive approach, the Apriori property enables the rejection of all subsets which are an extension of subsets that by themselves do not have sufficient support in the database.

Pattern-Growth Algorithms Another class of approaches avoids the expensive candidate generation aspect of the Aprior-type algorithms. FP-growth is one such algorithm, introduced by Han et al. [2000b]. This approach uses a divide-and-conquer technique. First, the database is transformed into a list of frequent ordered according to of descending support, and by key-criterion. From this list, a tree is constructed: a node is created for each frequent element, and they are arranged in the tree by their direct prefix item. If an element with an existing identical prefix is found, it is merged into the existing node and the frequency counter of that node is incremented.

From every node in this tree, it is then possible to identify frequent itemsets, by following a path of nodes toward the root, from each leaf corresponding to an item. Each such leaf defines a specific sub-itemset. The support of the itemsets are represented by the support of each leaf. A similar tree-based algorithm has been presented by Agarwal et al. [2001].

Vertical Data Structure Algorithms A third way to approach the problem of frequent itemset mining, is to take an orthogonal view at the database [Zaki, 2000]: Instead of assigning to each key criterion a set of items, assign to each item a set of elements from the key criterion. In an example, this would mean that instead of having a set of activities (items) assigned to each day (criterion), to instead assign to each activity the list of days where they occurred. Using this representation, the Apriori property can still be used to generate set extension candidates. Support for a candidate is determined by intersecting the sets that are created by the inversion of the database table. A key advantage is that determining support is as easy as counting the number of elements that are assigned to an item.

What is notable about this approach, is that the corresponding sequential pattern

mining algorithm SPADE [Zaki, 1998] was developed before this idea was ported to itemset mining.

Concept Hierarchy-Aware Algorithms An important extension to itemset mining is the acknowledgement that items may be related, and that at times it is useful to mine sets not only of items, but of classes of items. By using concept hierarchies, these classes can be established and (hierarchically) interrelated. One approach to mining such itemsets is a top-down specialisation approach, where first top-level general sets are mined, and then their specialisations are examined for sufficient frequency/support. This is possible if the minimal support threshold is constant across all abstraction levels of the concept hierarchy [Han and Fu, 1995, Srikant and Agrawal, 1995]. Han et al. [2006] present a variation of this approach to problems where minimal support varies by level.

Closed and Maximal Frequent Patterns Two further important concepts that restrict the scope of the problem to obtain more expressive results are the concepts of closed frequent patterns and maximal frequent patterns.

The former are patterns for which no pattern exists that encompasses it, where both have the same support. This eliminates from the results the most trivial subsets of frequent patterns, without reducing the overall information contained in the results compared to mining all frequent patterns. Any two patterns with different support remain separate.

Maximal frequent patterns on the other hand restrict this even further: all frequent itemsets that are subsets of another pattern that is frequent, are discarded. This means that the frequency information for these sub-patterns is lost. There exist Apriori-type algorithms optimised for closed itemsets (e.g. A-Close by Pasquier et al. [1999]) as well as FP-based algorithms (e.g. FPClose by Grahne and Zhu [2003]). Yang [2004] showed that enumerating maximal patterns is an NP-hard problem.

2.3.1.2 Sequential Pattern Mining

Sequential pattern mining, when compared to itemset mining, adds the additional dimension of (temporal) order to the problem. Itemset mining examines merely the co-presence of items with respect to the key criterion, whereas sequential pattern mining respects the order of appearance of the itemsets in the data.

A naive approach to pattern mining is to count the number of instances of every imaginable subsequence of a database (database being used synonymous with databank,

i.e. restricted to the physical storage of data). A subsequence in the context of sequence mining consists of itemsets, therefore the presence of multiple items in such an itemset produces ever more possible subsequences to test against the database. Therefore, much as for itemset mining, the goal remains to reduce the number of subsequences to test and the complexity of counting support.

Sequential pattern mining has relatively frequently been the subject of extensive surveys. Zhao and Bhowmick [2003] present some of the earlier developments in the field, Mabroukeh and Ezeife [2010] have established a taxonomy and cross-comparison of the key approaches to sequential pattern mining and more recently Mooney and Roddick [2013] have proposed a slightly differing classification and comparison on the same subject. Han et al. [2007] present a survey of approaches to the more general frequent pattern mining problem. All four surveys discuss in more detail what has been introduced here.

The following are the main approaches to sequential pattern mining, for the most part mirroring the concepts used in itemset mining:

Apriori-Based Algorithms On the base of the Apriori property, Agrawal and Srikant [1995] developed an Apriori-type SPM-algorithm, which has then been extended into the "Generalized Sequential Patterns" (GSP) algorithm in Srikant and Agrawal [1996]. This approach uses a similar minimal starting point, and then iterative growth of candidates, but with a fixed and predefined order for the sequential aspect.

An approach that integrates ideas from other algorithms is SPAM by Ayres et al. [2002]. SPAM generates a lexicographic tree and descends along the nodes in depth-first order. The Apriori property determines that any children of a node that does not have minimum support can be discarded from the search. Support is determined by counting incidences in a vertical data structure, which consists of a binary bit map of the sequence, upon which a binary **and** operation is executed.

Pattern-Growth Algorithms The divide and conquer approach has also been applied to sequence mining in the PrefixSpan algorithm [Pei et al., 2001, 2004]. PrefixSpan first determines the set of frequent patterns of length one, and then extends the patterns by determining possible prefixes from within this set, in a similar approach to FP-growth. To do this efficiently, the database is transformed into a more suitable representation. In the case of PrefixSpan, this is a projected database, as introduced in FreeSpan [Han et al., 2000a]. The projection mechanism for sequences and subsequences functions as follows: A sequence S' is a projection of its supersequence S with respect to a prefix P if there exists no other supersequence of S' that is also a subsequence of S which shares

the prefix P. This means that the extension of S' to S is done purely by extending the prefix P.

The database is rearranged according to the prefixes (which are the frequent items in the first step), where each prefix is assigned all occurrences of its postfixes by projection. By matching all postfixes with possible extensions of the prefix – to find frequent patterns – it is possible to determine a new set of prefixes upon which to project the database. A new scan if the database is not necessary, as the relevant sequences to project are already assigned to the prefix used in the previous step. This makes PrefixSpan fast, but the construction of the projected database becomes a major influence on the overall complexity of the algorithm.

Another Pattern-Growth approach uses tree projection instead of database projection. One representative from this group is WAP-mine Pei et al. [2000]. Similarly to the database projection approach, there are very few scans of the database - in this case two. The first determines the frequent items, the second builds the tree of frequent subsequences related to these items. The initial tree consists of an empty root, to which the frequent elements are added as nodes. The first element to be added is the first frequent item of the first sequence in the database. The second frequent item is added as a child. This is followed through for all frequent elements in their order of appearance in the first sequence. The same is done for the other sequences, but whenever an element is already present in the correct order in the tree, the existing element is used instead. In parallel, a header link table is established. It links each occurrence of an itemset to the next occurrences of the same itemset within the tree, beginning with the first.

To mine this tree, the least frequent item in the tree is chosen, and set as a *conditional suffix*. Based on this suffix, an intermediate tree is constructed, which consists of those branches that end (whether on a leaf or a node) on this suffix. These are identified by following the header links previously established. The new conditional tree therefore consists of the old tree, minus all the nodes header-linked to the corresponding suffix, and minus the branches that do not contain the suffix at all. For the next step, the suffix is grown by the least frequent item of the new tree, and the process repeated, until only a suffix and the root node are left. Each suffix generated this way is a frequent sequential pattern. This process is demonstrated in Fig. 2.3.1, with the sequences $abdac$, $eaebcac$, $babfaec$ and $afbacfc$, using the example from Pei et al. [2000].

Hybrid Algorithms SPADE [Zaki, 2001] can be considered as a hybrid approach, combining Apriori-based and Pattern-Growth characteristics. It uses an orthogonal view approach to itemset mining for sequential pattern mining. Similarly to Apriori-based

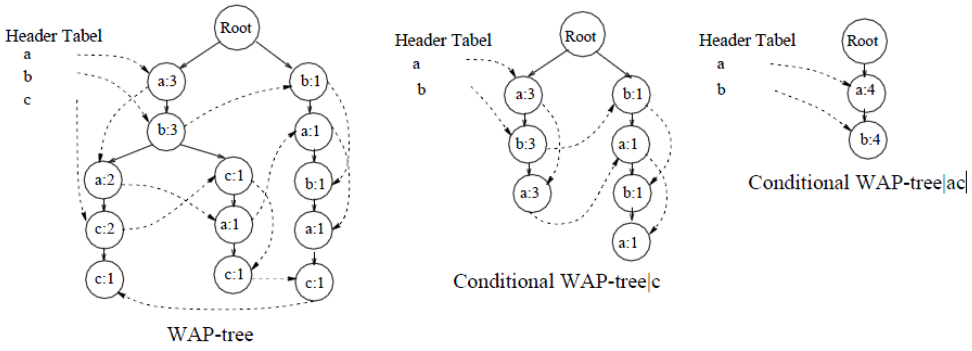

Figure 2.3.1.: A WAP-tree and conditional WAP-trees for two subsequences c and ac (Source: Pei et al. [2000])

approaches, this approach generates a large amount of candidate sequences that are then reduced by growing the length of the sequences progressively. SPADE transforms a database into a set of key-value pairs, where a list of pairs of sequences (in which it appears) and partition IDs of the key distinction criterion (e.g. a day) is assigned to each itemset (element of a sequence). The algorithm consists of three steps: first, the frequent sequences of length one are identified, then these sequences are extended to frequent sequences of length two. These are arranged in a lattice[1] structure, which is traversed in the third step, to count support and enumerate frequent sequences. This lattice structure can grow very large, as it defines an order across all frequent patterns. The authors propose to partition the lattice into smaller subsets, that are sufficiently self contained to be given the term *equivalence classes*. The strength of SPADE is mining maximal sequences, as when mining all frequent sequential patterns the performance is no better than GSP [Mabroukeh and Ezeife, 2010].

Early-Pruning Techniques This class of approaches exploits the sequential nature of the data, by tracking the positions of items that appear in the sequence. Specifically, if the (absolute) last position of an item is smaller than the position of the current sequence under consideration, then that item cannot occur behind this current sequence [Yang and Kitsuregawa, 2005]. LAPIN [Yang et al., 2007] maintains both a list of the last positions of frequent items, as well as a set of prefix border positions for each frequent item to achieve this. The former is a straight forward table, assigning each sequence a

[1] A lattice is a specific way to represent a (partially) ordered set, with operations defined to determine infima and suprema for pairs of set elements. See Davey and Priestley [2002] for a complete introduction.

list of indices corresponding to items that are frequent over all the database; the latter is generated from this list, and consists of all occurrences that have last appearances after the first instance of the item they are related to. By counting the elements in the second set – and determining whether they have sufficient support – it is possible to find the frequent patterns without searching the database.

Notable about this approach is that the data structure used is relatively static and has to be regenerated whenever new entries are added to the database. This makes determining the update frequency an engineering challenge when used in conjunction with high frequency data sources.

Closed Sequential Pattern Mining These basic approaches return *all* frequent patterns, particularly including all single elements that are frequent. In most applications this is undesirable, hence closed sequential pattern mining algorithms such as CloSpan [Yan et al., 2003] and BIDE [Wang and Han, 2004] have been developed. CloSpan exploits the property that when projecting a database of sequences with respect to two sequence prefixes, where one prefix is a subsequence of the other, the resulting two projected databases are equivalent if and only if the number of items of each of the projected databases is equal. This allows CloSpan to determine whether extending a sequence by an additional item creates an equivalent sequence or a sequence with different support. This permits the algorithm to distinguish between redundant candidate sequences and those that form a separate result. BIDE is an extension to this approach, that uses projections in two directions to remove non-closed frequent patterns.

This covers the most basic approaches to sequential pattern mining. With regard to the routine activity mining problem, there are two further characteristics of interest: multi-dimensional approaches and approximate approaches.

Multi-dimensional Sequential Pattern Mining The previously presented notion of "sequence" covers sequences of itemsets, which could comprise the information from multiple context data streams in parallel. Adding a clear notion of orthogonality between the different types of data allows for a more discerning treatment. By assigning each context data source its own dimension, there is no interaction between different states of different sensors.

The first "multi-dimensional" sequential pattern mining approach by Pinto et al. [2001] was so only with regard to allowing multiple key criteria according to which support is calculated. This allowed a multi-dimensional notion of support, but the actual itemsets where still restrained to consist of elements of a single dimension. This work has

been extended with an optimisation for closed multidimensional patterns in the work of Songram et al. [2006] and Boonjing and Songram [2007].

Plantevit et al. [2010] undertook the next logical effort, to treat sequences of multi-dimensional, multi-level data. They develop the bottom-up approach described by Beyer and Ramakrishnan [1999] to mine maximally atomic frequent sequences, which are sequences consisting of a single (hence atomic) multi-dimensional item. This item must be the most specific – with regard to the concept hierarchy model – item that is still frequent in the database. From this point, the sequences of length one are built up to longer sequences using the SPADE algorithm [Zaki, 2001].

Approximate Sequential Pattern Mining The realisation that exact pattern mining is too restrictive for real world problems has been reached by Wang et al. [2000]. The first approach to mining long sequential patterns from noisy data by Yang et al. [2002] uses a probabilistic model to account for the noise: a *compatibility matrix* consisting of the conditional probabilities, that x is the real value, given that y has been observed, gives a notion of relative proximity of values, with regard to the measurement (or data generation) process. The notion of support is adapted to become noise-tolerant. The actual significance criterion under noisy influences on the data is termed *match*, and is defined as the maximum conditional probability of an occurrence of a pattern, given any one subsequence of a sequence, averaged across the database. A slightly modified Apriori-property holds for the match measure as it does for support, if subpatterns are defined as a pattern that lacks some symbols present in the respective superpattern.

Although theoretically it is possible to use classical algorithm with this model, the large number of candidates and database scans generated using Apriori-/support-based approaches makes them impractical. Instead, the authors propose a statistical filtering approach to reduce the number of passes, which is based on two criteria: First the additive Chernoff bound [Hoeffding, 1963, Domingos and Hulten, 2000] is used to estimate the size of a pattern (i.e. the points beyond which extending a subsequence reduces its match below the relevance threshold) based on the results of random sampling. This divides the set of candidates into three (probable) classes: frequent, infrequent and ambiguous patterns. The second optimisation deals with the ambiguous patterns – which require further attention – to determine the frequent and infrequent patterns contained within. The specific approach refines the borders (left and right limits of the end of the ambiguously frequent subsequence) obtained during the random sampling approach by collapsing them down to the actual borders. This is done using a hierarchical division of the two estimated borders, which takes at worst $O(\log(n))$ steps to find the correct

border, where a classic linear algorithm would take n steps. Overall performance of the approach depends very much on the data and the number of ambiguous patterns that cannot be identified during the first probabilistic step.

ApproxMAP [Kum et al., 2003] – another approximate method – borrows some notions from string mining. This algorithm mines *consensus patterns* (i.e. short patterns that appear in similar fashion across many of the examined sequences) through multiple pattern alignment. It uses a hierarchical edit distance as a similarity measure for pairs of sequences, and thus determines clusters of similar sequences. In a second step, a representative for each cluster is selected. A consensus pattern is determined to be present whenever a sufficient number of sequences in the cluster share this representative to achieve a *strength threshold* – a notion similar to "minimum support".

Applications in Context Pattern Mining The main application to context data mining is the field of trajectory mining and more generally spatio-temporal data mining. Due to the limited scope of this problem, compared to the routine activity detection problem, we only briefly introduce a few key approaches.

Rashad et al. [2007b] developed a multi-dimensional PrefixSpan [Pei et al., 2001] specialisation called MobilePrefixSpan (based on the work of Pinto et al. [2001]), to mine movement patterns of mobile users. Their database consists of entries, describing which user was present in which mobile phone network cell, at which time. Using the generated movement profiles, they try to predict future positions of users, with the goal of providing better resource management in the wireless network itself. Compared to the original PrefixSpan, they restrict sequences to consecutive sequences, as they argue that the exact order in which cells are visited is more important than the more global trends that traditional sequence mining detects.

Giannotti et al. [2007] also perform pattern mining on GSM or GPS location data. They extend the basic SPM model with annotations indicating the typical transition times between elements (see also the previous work of Yoshida et al. [2000]) and replace itemsets by spatial points. The calculation of support is not done by exact matching, but instead by an error tolerant neighbourhood function. One of the key problems of this approach, is that of determining *Regions-of-Interest* – i.e. semantically similar regions, to determine the neighbourhood function – for which they provide a seed-and-growth approach, using popular spatial points as starting points. The actual mining algorithm is introduced in Giannotti et al. [2006] and uses prefix projection (like PrefixSpan [Pei et al., 2001]), but modified to accept time-stamped sequences.

Further work in this field can be found in the works of Kang and Yong [2010], Lei and

Wong [2009], Nanni et al. [2010], Zhao et al. [2013], Zheng et al. [2011].

2.3.1.3 Discussion

SPM, at first glance, presents a suitable approach to the routine activity detection problem. The blocking metaphor (selecting sequences from a database by a key criterion) works well with the notion of days and weeks structuring human activity. The performance is compatible with the volumetry of context data (1). Although actual algorithmic complexity largely depends on the data, the sample data presented usually scales up to hundreds of thousands of itemsets in the sequences. The notion of closed patterns and maximal patterns allow refining the result to those patterns that should be the most interesting.

The traditional SPM is limited, in that it can only identify exact patterns. Semantic hierarchical models can help with this (3,6). In this case, similar sensor values are grouped under a more general label, to allow for more general patterns to be mined. Additionally, approximative approaches exist, that are able to find patters in noisy data, or use string mining approaches to identify similar sequences, thus addressing criterion (4).

Although these hierarchical approaches are more realistically employed to solve the routine activity detection problem, there still persists one problem. Treating each sensor merely as one source of items that are all fundamentally equivalent, in that in theory each item can be replaced by any other, leads to a large number of comparisons that can be prevented, if from the start the data is considered multidimensional. Plantevit et al. [2010]'s multilevel and multi-dimensional approach is capable of eliminating these ambiguities, which should in theory allow an optimized treatment of such truly parallel data. On the other hand, this approach is limited to exact matches, which puts higher demands on the preprocessing of the context data. As with the other approaches, the non-consecutive sequences that do not make sense need to be removed from the results. Lastly, tuples (i.e. multi-dimensional datasets) are treated as a unit, so desynchronised context data would prove problematic (5).

A notable drawback then, across almost all SPM algorithms – when dealing with context data – is that in the classical retail shopping scenario, items are numerous and sequences are short, but for the context scenario, sequences are long, and "items" are relatively few. This is the foremost limitation, as non-consecutive patterns can appear by coincidence much more easily in these conditions, without actually indicating a recurring activity. The consecutive approach by Rashad et al. [2007b] provides a solution to this, but inversely the increased requirement for exactness (as now erroneous values are not

skipped) requires more intense preprocessing, which renders an adoption problematic. The time-interval supporting approaches by Yoshida et al. [2000] and Giannotti et al. [2006] allow to quantify these intervals between items and use them to determine patterns that take these intervals into account.

Compared to SPM, string mining addresses approximate approaches much more comprehensively, and with less inherent complexity.

2.3.2 String Mining

Strings are finite length character sequences. By assigning to each state of a context or sensor a specific character, it is possible to represent context data sequences with strings. String mining, as a discipline, is closely linked to bioinformatics and computational biology. With the discovery of the structure of DNA, RNA and protein structure, and the ability to obtain base pair sequences and amino acid sequences from cells, the search for meaning in these sequences became a major research topic. This so-called "biological revolution" has motivated a large parallel effort in the development of effective and efficient sequence analysis (i.e. string mining) algorithms.

In their survey of string mining in bioinformatics, Abouelhoda and Ghanem [2010] provide a taxonomy of the field, discerning principally between repeat-related problems and string comparison problems.

Repeat-related problems try to find repeated or otherwise interesting subsequences within larger sequences, whereas string comparison problems are based on comparisons between two separate strings, and often inspired by a need to determine a similarity metric. Each of these two categories is explored in the following, with a closer look at approaches that are of relevance to our routine activity detection problem.

2.3.2.1 Repeat-Related Problem

Repeat-related problems can be divided into the following subclasses (c.f. Fig. 2.3.2):

Dispersed Repeats i.e. finding repeats through a sequence. These repeats can be either approximate or exact, with fixed or variable length.

Tandem Repeats i.e. repeats that occur in an adjacent manner. These repeats can also be either exact or approximate in nature.

Unique Subsequences i.e subsequences that do not have repeats and appear only once in a sequence.

Absent Words i.e. subsequences that do not exist at all in the original sequence. Only the shortest absent words are of interest, as generally there is an infinite number of subsequences not contained within a sequence.

With regard to the class of repeat-related problems, our task falls within the scope of a dispersed longest repeat problem: Find a pair of sub-strings that match, where neither the pairs of preceding elements nor the pairs of succeeding elements match. Alternatively, if a sufficiently powerful classifier were to exist, which could consistently assign a correct activity label to a set of sensor readings, the routine activity detection task could be expressed as an exact repeat problem; otherwise it would be a more general approximate repeat problem. The approximate repeat problem is solved by a local alignment algorithm in $O(n^2)$ [Smith and Waterman, 1981], which technically belongs to the class of comparison problems, and is described in the corresponding – next – subsection.

A naive, brute-force approach to finding (longest) pairs of exact repeats would be to create a scatter plot, and scan along its diagonals for series of matches. Given n lines of input, this type of approach would use $O(n^2)$ time and $O(n)$ space.

Figure 2.3.2.: Examples of different types of repeat-related features in a string.

Suffix Tree Approach A suffix tree [McCreight, 1976] based approach appears more suited for large volumes of data. Baker [1992] proposes an exact matching algorithm using the suffix tree structure, that finds pairs of maximal repeats in a sequence of length n and m repeats in $O(n + m)$ time. It follows a description of this approach and an illustration on the example input string *abcbcabc*% (% is the string termination symbol).

The algorithm consists of four steps:

1. A suffix tree is generated, as follows:

   a) An empty tree is generated (as in Fig. 2.3.3-0);

   b) A leaf containing the entire string (appended with a termination symbol "%") is created (as in Fig. 2.3.3-1);

   c) A new leaf containing the first suffix (comprising all but the first element) is created (as in Fig. 2.3.3-2 to -6, and Fig. 2.3.4 for the final three leaves);

   d) If the head (i.e. the first elements of this suffix) is present in an existing leaf, a new node representing this head is created, and two leaves representing each of the possible suffixes to this head are added to it (e.g. in Fig. 2.3.3-4 for *bc*, when the suffix *bcabc*% is added to the tree which already contains a leaf *bcbcabc*%);

   e) Repeat (c) and (d) with the first suffix of the remaining string (as in Fig. 2.3.3-3 to -6 and Fig. 2.3.4 for the final three leaves, 7, 8 and 9). Each newly added leaf representing a new suffix is assigned the index of its first element in the original string.

2. In the suffix tree, identify head nodes (circular nodes in Fig. 2.3.4) which satisfy a minimum length condition (≥ 2 for this example). They correspond to repeating sub-strings, with differing right elements (as otherwise the head node would already comprise this right element). The left element may still be identical. In the example these are the nodes *bc* and *abc*.

3. Build a list of suffixes, grouped by left elements: For each possible element to the left of the node "head" element, a list is created. Each leaf is added to the corresponding list. In the example, this is shown for node *bc* in Fig. 2.3.4, with the left elements *a* (in the case $*\underline{a}bc*$, leaves 2 and 7) and *c* (in the case $*\underline{c}bc*$, leaf 4). The leaves are emphasised by being enclosed within the dashed ellipses, and labelled with the prefixes.

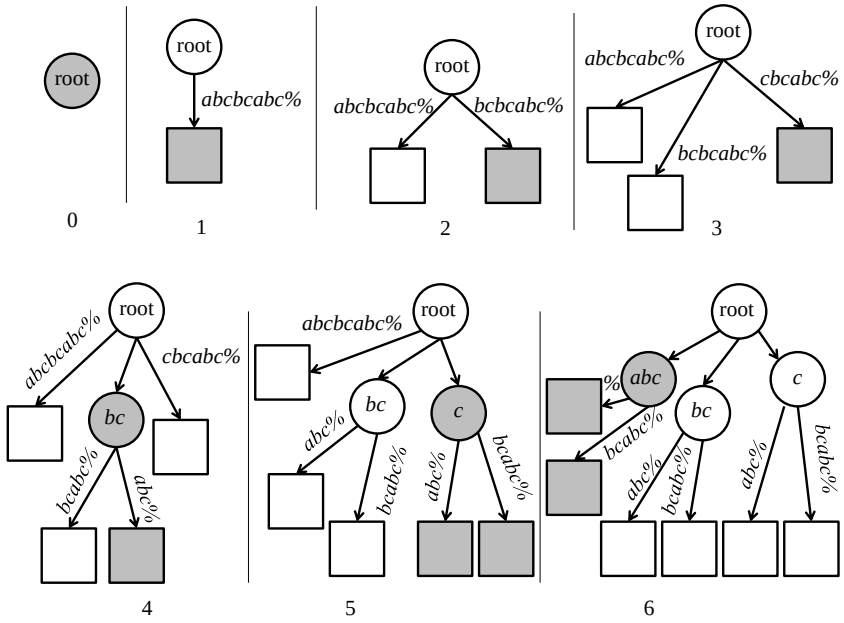

Figure 2.3.3.: Construction of a suffix tree from *abcbcabc%*. The numbers indicate both the order and the index of the current suffix being added. New leaves and nodes at each step are shaded.

4. Form the cross-product of each distinct pair of lists for all nodes. In the example this is the cross-product of the lists $(2, 7) \times (4) = (2, 4), (7, 4)$ for the node *bc*, meaning that *bc* is the longest repeat for the instances of *bc* in the fourth position in the string, with the instances in the second and seventh position.

The pair of leaves 2 and 7 of the node *bc* in the example are *not* longest matches, as the preceding character is an *a* each time. Therefore the actual longest match is to be found elsewhere: in the node *abc* and the pair of leaves 1 and 6.

Applications to User Modelling Within the context of routine activity detection, this approach has been used by Pitkow and Pirolli [1999] to mine users' web access logs, with the goal of predicting websites that a user will visit in the near future. After extraction of the longest matching subsequences, these subsequences are split, and Markov models (of different orders) are used to obtain transition probabilities. The key use of the longest repeat problem in this case was to optimize performance over a previous approach, by limiting the generation of Markov models to the data contained within longest matching subsequences, while maintaining similar prediction performance.

Similarly, Pauchet et al. [2009] use suffix-tree based repeat-mining to identify recurring

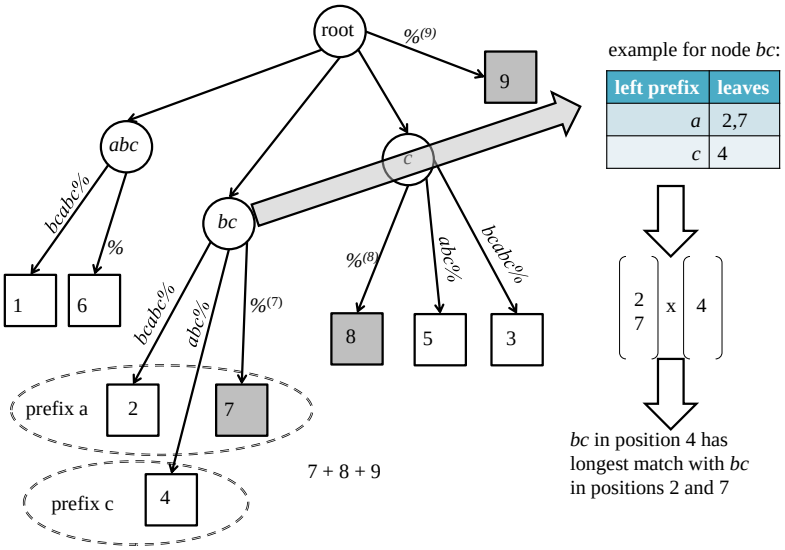

Figure 2.3.4.: Identification of longest repeats using Baker's algorithm in the complete suffix tree from Fig. 2.3.3. Steps 7, 8 and 9 of the tree construction are performed, adding the three shaded leaves. The order in which the leaves are added is given by the nodes and indicated by the superscripts of the vertex labels.

behaviour of users of medical catalogues. This information is then used to present a subset of works from the catalogue to the user, which contains elements which are expected to be part of the desired search result.

In the field of pervasive and mobile computing, Katsaros and Manolopoulos [2005] propose an approach to – for example – track the location of a client in a wireless cellular network. They describe a prediction task, and also use a suffix tree to find longest exact matches, from which they construct a probabilistic model of transitions.

Sigg et al. [2010] use exact (or approximate) repeat search as the first stage of their context prediction algorithm. This stage is used to determine "typical" contexts. Suffixes of the observed context are then aligned with these typical contexts, to find the best match. They then predict the next context to be the continuation of the identified typical context.

2.3.2.2 String Comparison Problem

String comparison problems can be categorized as follows:

global i.e. compare entire strings;

semi-global i.e. search problems – finding short strings within a larger one;

local i.e. problems on sets of sub-strings. Local problems can further be subdivided into problems with variable length and fixed length of these sub-strings.

Furthermore, each of these problems has *exact* and *approximate* variations.

When approaching the task of identifying routine activities from context data as a comparison problem, then this problem should be interpreted as local comparison problem, either of exact or approximate nature, and of variable length. Additionally, a dispersed approximate repeat problem can be reformulated in terms of an equivalent comparison problem: comparing pairs of subsequences of the sequence in which approximate repeats are sought.

Exact Matching Local exact matches (i.e. identical sub-strings within two larger strings) can be found in a brute force way: Pair all sub-strings of one string of length n and the other, and check for identity. Due to the identical length requirement, this implies $O(n^3)$ string comparisons. Search space reduction to strings of length l further reduces this to $O(n^2 l)$.

To render the problem more tractable, it is possible to limit the search to maximal exact matches. In this case, Baker's suffix tree algorithm for finding maximal repeats (see subsubsection 2.3.2.1) can be modified to find maximal exact matches instead. The modifications are the following: The lists of positions (shown in the right part of Fig. 2.3.4) are split into two subsets, one containing those suffixes belonging to the first of the input strings, the other containing those that belong to the second input string. The result is then obtained by forming the Cartesian product of each pair of lists, where both the input string and the left element are distinct. This minimal modification has no effect on algorithmic time and space complexity.

A complete survey of a large number of both recent and early approaches to exact on-line string matching (i.e. finding a known pattern in a string) can be found in a review of the field by Faro and Lecroq [2013]. Each of the algorithms in the survey has been tested against a battery of synthetic and real world data, and characterised in how well it suits two problem characteristics, pattern size and alphabet size. We voluntarily omit detailed study of these approaches, as expressing the routine activity detection problem as a search problem is highly inefficient, due to the combinatorial scale of the possible, unknown, patterns. The identification problem ("Is the current context a known context?"), which these algorithms address, is much less complex than the routine mining problem, and even a worst case $O(n)$ algorithm [Knuth et al., 1977]

has little impact on the overall performance.

Applications to User Modelling One application of this algorithm in the field of context data, is the SHIP algorithm by Cook et al. [2003], which uses exact matching to determine frequencies of following activities, with the goal of predicting future context from recognising frequent sequences. This approach can also be classified as a sequential pattern mining approach, given the fact that the exact matching is used to extract a frequency measure for a part of a sequence. The approach requires permanent access to a history of past activities, in order to perform a matching of the current subsequence with historic data. SHIP is one of multiple algorithms of the MavHome smart home architecture. This is an agent-based multilayer architecture, which uses a predefined set of concepts to transform physical sensor data ("lowest" layer) into abstract, discrete context data, before it is treated by the learning and decision-making modules ("highest" layer).

String matching algorithms have also been adapted to two- or multidimensional data. This is possible through an extension of the string-matching paradigm to "wider" data structures (i.e. where each string element itself is a string of length greater than one) which has been proposed by Baker [1978]. The problem is reduced back to a string matching problem, which allows the use of efficient, well known algorithms. First rows from the search pattern are matched with rows from the subject array, then a table of matches annotated with a row ID is created. In this table, the row IDs are matched with the order of row IDs in the pattern array, column-wise. Therefore, the algorithm effectively represents the problem as the concatenation of two string matching problems.

A similar approach is taken by Zhu and Takaoka [1989]. They use the hashing pattern matching approach of Karp and Rabin [1987] on the columns of the input data to first reduce the array problem to a string problem, and then use the algorithm of Knuth et al. [1977] row by row to find the array patterns.

To the best of our knowledge, these approaches have not been used within the field of context data mining or applied to related fields, but could be considered as candidates, given a suitable framework.

The k-Error Problem The k-error problem (explored in depth in the work of Navarro [2001]) is a quite specific approximate string comparison problem, where the goal is to match a pattern to a string, without incurring more than a fixed number of modifications, and the more general alignment problem, in either global or local flavour. A global alignment of two strings is the ideal ordered set of operations to transform one string to

another, whereas local alignments are based on the same principle, but look instead to find maximally similar sub-strings within the pair of input strings.

The k-error problem is too limited in scope, to be of use in the simple string case, as it is restricted to a search functionality, and the constraints of the fixed number of changes make no sense when dealing with context data. Nonetheless, further on we cover some variants of this approach, that generalize it to two dimensions, where the reasonable complexity of the solutions has some more interest. The class of alignment problems on the other hand is more flexible, and thus more interesting of analysis, and can also be used to solve the k-error problem.

Global Alignment The first foray into the field was led by two biologists, Needleman and Wunsch [1970]. They propose a method to find the largest similar subsequence of a pair of amino acid sequences. More specifically, their algorithm finds similar subsequences of maximum length. Their original algorithm – which has largely been surpassed by a dynamic programming version – is illustrated for an example in Fig. 2.3.5. Two input strings are orthogonally arranged, so that a table can be spanned between them. In this version, every match of two symbols is given a score (1) in a table – as shown in the left table in Fig. 2.3.5. This value is added to the largest value in the top left sub-table – the empty table is assumed to contain the score 0 – this process is detailed in the right table of Fig. 2.3.5 . Following the increases in score across the table, gives the operations required to perform an alignment (denoted by arrows in the example).

Non-deterministic configurations can be encountered, for example when one string contains a sub-string in inverse order as the other string (e.g. *abcd* and *acbd*). In that case, the following value can be based on either of the two previous rows of the table. This is because the incremented value for the b-b-match is *not* in the sub-table from which the maximum is chosen when examining the c-c-match. This leads to two equal values, one in each row of the table, and a common value in the following table. Because the top-most of the equal values is to the right of the lower-most, it is not possible to traverse both values, leading to the non-deterministic situation, where one of the two equally valid paths (deletion of b or deletion of c) has to be chosen.

Dynamic Programming Algorithm The formalisation of the dynamic programming algorithm for global sequence alignment [Sankoff, 1972], requires the definition of a few terms. Let Σ be an alphabet of characters and let Σ^* be the set of all possible combinations of characters into sequences. Let ε be the empty word. Let $a \in \Sigma$

	A	C	G	T	C	G	A	C	G
A	1						1		
C		1			1			1	
T				1					
C		1			1			1	
A	1						1		
C		1			1			1	
G			1			1			1

	A	C	G	T	C	G	A	C	G
A	**1**	0	0	0	0	0	1	0	0
C	0	**2**	1	1	2	1	1	2	1
T	0	1	2	**3**	2	2	2	2	2
C	0	2	2	2	**4**	3	3	4	3
A	1	1	2	2	3	4	**5**	4	4
C	0	2	2	2	4	4	4	**6**	5
G	0	1	3	2	3	5	4	5	**7**

Figure 2.3.5.: Original Needleman-Wunsch algorithm, as introduced by Needleman and Wunsch [1970]. The left table is filled with incrementation markers ("1") at each location where a pair of values match. In the right table, the values for all the cells have been determined, and the alignment highlighted by arrows and bold values. The alignment (arrows) starts at the maximum value and then follows the maximum values, upwards and to the left in the table. The "skips" between values 5 and 4, and 3 and 2 are indicative of deletions in the row-string.

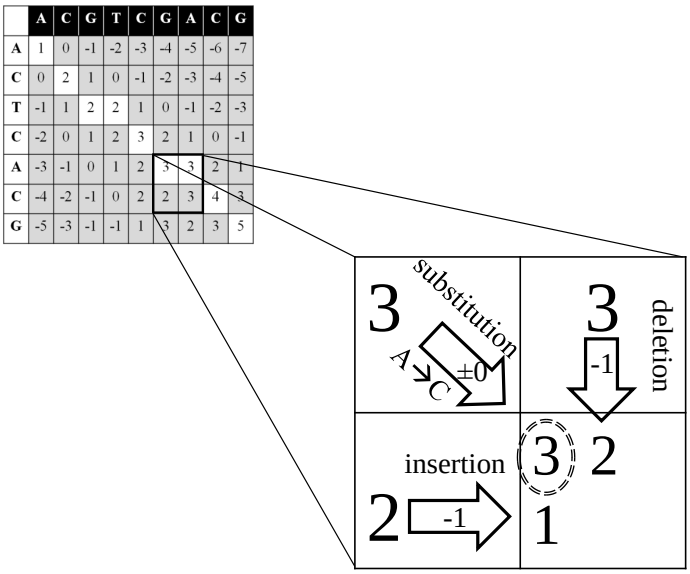

Figure 2.3.6.: Calculation of the local similarity using the dynamic programming approach. The preceding values in the table are either incremented by the substitution score, in the case of the diagonal predecessor, or decremented by the indel penalty. The maximum value is carried over for all successive operations.

and $b \in \Sigma$ $(a \neq b)$ be two such characters. For each pair of (a, b), define an element-wise replacement edit distance or substitution distance $\text{sub}(a, b) \in \mathbb{R}$ where it holds $\text{sub}(a, a) \neq \text{sub}(a, b)$. In the original paper, the proposed values were 1 for a match and 0 for a mismatch. An alignment can comprise insertions and deletions (*indels*). These operations are assigned a penalty value $\text{indel} \in \mathbb{R}$ $(= -1$ in the original paper).

The next step is the accumulation of these values in an array ("accumulated similarity score table") spanned by the two input sequences. Row by row, the array is filled. Each cell in the array is calculated as the maximum of the sum of values in the directly adjacent cells with already determined values and the indel penalty, and the sum of the value in the diagonally adjacent cell with already determined value and the appropriate substitution score for the position of the cell (cf. Fig. 2.3.6). Formally: For two sequences S_1 and S_2 with last elements a and b respectively, the similarity score $\text{sim}(S_1, S_2)$ is recursively defined as:

$$\text{sim}(S_1, S_2) := \max \begin{pmatrix} \text{sim}(S_1^{-1}, S_2^{-1}) + \text{sub}(a, b) \\ \text{sim}(S_1^{-1}, S_2) + \text{indel} \\ \text{sim}(S_1, S_2^{-1}) + \text{indel} \end{pmatrix} \tag{2.3.1}$$

where S^{-1} is the prefix of length $|S| - 1$ of a sequence S. The initial condition is $\text{sim}(S, \varepsilon) = \text{sim}(\varepsilon, S) = 0$. Once the array has been completely populated, the maximum value in the last row or column is selected as origin. From this point, a trace of maximum values in the array is followed in the opposite direction of its generation. The direction taken at each step corresponds to an alignment operation. A diagonal movement corresponds to a substitution, whereas a movement into a directly adjacent cell corresponds to an indel. The sequence of operations that is returned this way is the sequence of operations required to align one of the input strings with the other.

Using a dynamic programming approach, the algorithm can be implemented within the constraints of $O(n^2)$ time and space. When comparing k sequences (of average length n), the complexity is of the order of $O(n^k)$.

Optimal Global Alignment Hirschberg [1975] proposes a linear space algorithm, which computes only the optimal score, and returns only the optimal sequence of alignment operations. This is achieved through a divide and conquer approach. Let S^i be the suffix consisting of the last i elements of a sequence S and S^{-i} be the corresponding prefix.

Given sequences $S_1, S_2 \in \Sigma^*$, with $|S_1| = n$ and $|S_2| = m$, it holds:

$$\forall_{0 \leq i \leq n} : M(i) := \max_{0 \leq j \leq m} \{\text{sim}(S_1^{-i}, S_2^{-j)}) + \text{sim}(S_1^i, S_2^j)\} \rightarrow M(i) = \text{sim}(S_1, S_2)$$

This means that, for each position i in S_1, there exists a position j in S_2 such that the sum of similarity values of the pair of prefixes and the pair for suffixes from these positions is equal to the similarity value of the pair of sequences. Splitting S_1 approximately in half (at index i) therefore implies an optimal split of S_2 at the j where the minimum $M(i)$ is calculated. Repeat this process on the pairs of first and second subsequences generated by the split until only a trivial problem (such as S_2 being empty, or S_1 containing only a single symbol and S_2 being not empty) remains.

The dynamic programming approach is illustrated – for the same pair of strings as in the previous example – in Fig. 2.3.7. Note how the value in each cell depends only on the three neighbouring values (as indicated in equation 2.3.1), which is crucial to Hirschberg's approach.

Local Alignment Global alignments are of interest, when a corpus of known interesting subsequences exists. As referenced above, Sigg et al. [2010] use global alignment

	A	C	G	T	C	G	A	C	G
A	1	0	-1	-2	-3	-4	-5	-6	-7
C	0	2	1	0	-1	-2	-3	-4	-5
T	-1	1	2	2	1	0	-1	-2	-3
C	-2	0	1	2	3	2	1	0	-1
A	-3	-1	0	1	2				
C									
G									

±0 -1 -1

	A	C	G	T	C	G	A	C	G
A	1	0	-1	-2	-3	-4	-5	-6	-7
C	0	2	1	0	-1	-2	-3	-4	-5
T	-1	1	2	2	1	0	-1	-2	-3
C	-2	0	1	2	3	2	1	0	-1
A	-3	-1	0	1	2	3	3	2	1
C	-4	-2	-1	0	2	2	3	4	3
G	-5	-3	-1	-1	1	3	2	3	5

Figure 2.3.7.: Dynamic programming version of the Needleman-Wunsch algorithm. Non-matching substitutions have no penalty, matching substitutions score a value of 1 and insertions and deletions are penalised with a value of -1. The left table shows how a particular cell value is calculated from three predecessor values, the right table shows the complete table and highlights the series of maximal values which indicate the alignment operations. Note that the traceback follows substitutions over indels in cases of ambiguity (e.g. the 4→3 transition marked in the right table), as the value that was used to calculate the current value defines the path for the traceback, and not necessarily the maximum value.

to identify known context sequences in a stream of real-time context data. In order to actually identify such repeating patterns in sequences, local alignments need to be found.

The reference local alignment algorithm is Smith and Waterman's algorithm [Smith and Waterman, 1981], a variation on the dynamic programming Needleman-Wunsch optimal matching algorithm described earlier. The key differences are that negative values are used to represent the similarity of different values, and that in the accumulated similarity score table all negative values are truncated to zero. The similarity function above is modified to:

$$\mathrm{sim}(S_1, S_2) := \max \begin{pmatrix} 0 \\ \mathrm{sim}(S_1^{-1}, S_2^{-1}) + \mathrm{sub}(a, b) \\ \mathrm{sim}(S_1^{-1}, S_2) + \mathrm{indel} \\ \mathrm{sim}(S_1, S_2^{-1}) + \mathrm{indel} \end{pmatrix} \tag{2.3.2}$$

and furthermore, a backtrack halts when a zero value is encountered. The result is that – in the accumulated similarity score table – the score rises along pairs of local subsequences that are similar, then drops as they become dissimilar further on, and finally reaches zero. This zero value then serves as a delimiter: any similarity elsewhere is not impacted by the previous values, and thus locality is introduced. The degree of locality is directly dependent on how quickly similarity scores are accumulated to reach zero, from the point that the two subsequences diverge. This means: the dissimilarity of a pair of intervals between two pairs of similar intervals determines whether the two intervals are treated as a single pair of similar sub-strings containing the dissimilar sub-strings, or as two separate pairs.

An example of this is given in Fig. 2.3.8, with two different penalties (-2 and -1 for left and right tables respectively) used to calculate the two tables. The result is that the smaller alignments are each time reset (to a zero score) in the left table, before they can form a larger alignment.

Although Myers and Miller [1988] showed that it is possible to apply Hirschberg's approach to local alignments, this is not always desirable, as reducing the result to the optimal alignment, discards all other local alignments present in the data.

Fast Search Algorithms In bioinformatics, one of the key challenges is to check for the presence of medium length sequences within a genome. This is best achieved by algorithms that are optimized for search, such as FASTA by Lipman and Pearson [1985] and BLAST by Altschul et al. [1990]. These algorithms use heuristics to achieve

	A	C	G	T	C	G	A	C	G
A	1	0	0	0	0	0	1	0	0
C	0	2	0	0	1	0	0	2	0
T	0	0	0	1	0	0	0	0	0
C	0	1	0	0	2	0	0	1	0
A	1	0	0	0	0	0	1	0	0
C	0	2	0	0	1	0	0	2	0
G	0	0	3	1	0	2	0	0	3

	A	C	G	T	C	G	A	C	G
A	1	0	0	0	0	0	1	0	0
C	0	2	1	0	1	0	0	2	1
T	0	1	2	2	1	1	0	1	2
C	0	1	1	2	3	2	1	1	1
A	1	0	1	1	2	3	3	2	1
C	0	2	1	1	2	2	3	4	3
G	0	1	3	2	1	3	2	3	5

Figure 2.3.8.: Two accumulated similarity tables obtained using the Smith-Waterman algorithm. The left has been calculated using a similarity score of 1 for matches, and dissimilarity penalties of -2 for non-matching substitutions and indels. The right table has this penalty reduced to -1. In each case, the alignments with a similarity score of at least 3 have been highlighted. Note how the higher penalty leads to smaller, more local alignments.

much better search performance than exact methods, at the cost of a guarantee that the obtained results are correct. By being limited to search, these approaches do not offer themselves to the more exploratory nature of the routine activity detection problem pursued in this work. A further number of approximate on-line string matching algorithms is exposed in a survey by Navarro [2001], which introduces algorithms that are also based on statistical approaches, automata based approaches, filtering approaches and bit-parallelism based approaches.

Two-Dimensional Pattern Matching Approximate pattern matching approaches for multiple dimensions can be divided into two classes: error-tolerance-based approaches and alignment-based approaches.

The former use a simple model, that merely counts the number of modifications required to transform one structure into the pattern that is being sought. If the number of operations required exceeds a limit, a mismatch between data and pattern is assumed, similar to the k-mismatch approach for strings. Krithivasan and Sitalakshmi [1987] present a row-based approach to this problem, and a simple generalisation of the dynamic programming algorithm. Their model of a pattern is rectangular. The algorithm consists of two steps: first patterns are represented in an optimized fashion, by reducing rows that are identical or similar to a differential representation. Then, the data is searched for occurrences of the first row of the pattern, and for each occurrence it is

verified whether the second row of the pattern follows. The number of required modifications is tracked for the set of occurrences, and whenever the error-limit is exceeded, the occurrence is discarded.

Amir and Farach [1991] present an early algorithm that takes into account general structures, as opposed to merely rectangular ones. To achieve reasonable run time, they use numerical convolutions to perform the approximate matching. Baeza-Yates and Navarro [1998] present an approach of identical optimal complexity, that consists of a filtering step, discarding all rows in the text that cannot possibly contain a pattern, before using a standard dynamic programming algorithm for the final matching.

Two-Dimensional Motif Extraction The previously introduced two-dimensional matching approaches do not allow the extraction of patterns from data, but instead solve the problem of finding a known pattern within data. As in the one-dimensional case, matching alone is not the key issue of the routine activity detection problem. Instead, the identification of common patterns in input data, is the key problem of the routine activity detection problem. Such extraction algorithms have first been proposed by Apostolico et al. [2008]. The notion of a pattern in their work is still essentially rectangular, but through the use of "*don't care*"-symbols (symbols in patterns that match any symbol in the input data) the actual informational content of a pattern can take arbitrary shapes. Their approach primarily searches for autocorrelations of a single input array. An autocorrelation in their nomenclature is a similarity between the array and its transposed array. They propose an incremental combinatoric approach to detecting a *base* of patterns. A base is a set of maximally sized and maximally dense patterns, which comprises all patterns of an autocorrelation. Within the framework of the routine activity detection problem, such a base would correspond to a set of recurring contexts of maximal length and specificity, that describes all recurring contexts in a specified time frame.

The incremental algorithm functions as follows: iterating over the cells of the array, in a row-major order, from the lower right corner to the upper right, a base is found for each sub-set of elements contained up to the current position. During the iteration step, new base element patterns are generated, through inclusion of the new symbol. Some of these patterns are novel – i.e. have not been generated previously – and render some old patterns obsolete, either by rendering them more specific, or by extending them in size. New patterns can be not novel, for example when a novel pattern removes an old pattern through extension from the base, but a previously removed pattern does then again become a valid base element, through this removal. Iterating this process across

data of size $N = m \times n$ requires $O(N^3)$ time. By reducing the alphabet to a binary alphabet, complexity can be reduced to $O(N^2)$ [Rombo, 2009].

Two-Dimensional Local Alignment A generalizations of the local alignment by dynamic programming paradigm to two dimensions is presented in Lecroq et al. [2012]. This approach can be seen as an extension of the Smith-Waterman algorithm. It computes similarity scores for each pair of "prefixes" (in the two-dimensional case, this is the array to the top and left of the chosen position) in a table, that has now gained four dimensions. Two of these dimensions correspond to the widths and the other two dimensions correspond to the lengths of the two arrays being locally aligned.

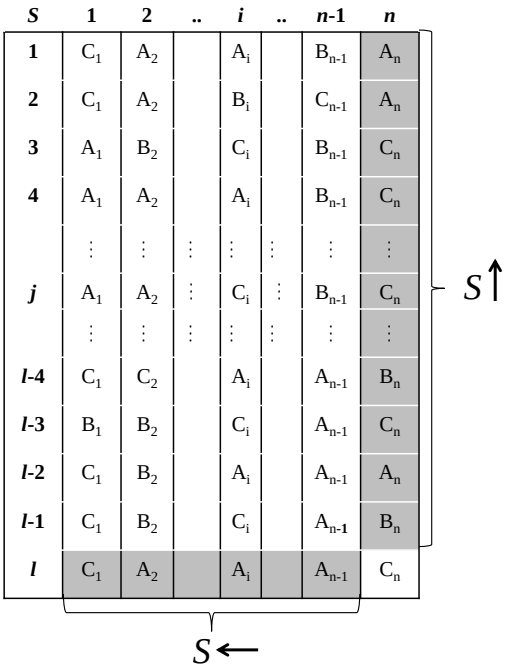

Figure 2.3.9.: Row and column prefixes, and accompanying notation.

The similarity value in the table entry is computed as a function of the preceding values, much as in the Smith-Waterman dynamic programming algorithm. The difference in the two-dimensional case is that the preceding values now number eight instead of three. First of all, the number of possible deletion and insertion operations has doubled (lines V to VIII in equation 2.3.3), as they can be either in vertical (VII, VIII) or horizontal (V,VI) direction, and furthermore substitutions can be accompanied by different movements in the top-left direction. These directions are either top (IV), left (III) or top-left (in both arrays simultaneously - I and II). Finally, in the case of the top-left movement, the order of the sub-movements (top-left (II) or left-top (I)) influences the score. This is the case, because row prefix and column prefix similarities play a role in the calculation of the substitution scores, taking the place of the per-element substitution scores used in the one-dimensional case. In the following, let $S \leftarrow$ and $S \uparrow$ denote the row prefix and column prefix of the bottom-right element of a sequence S (cf. Fig. 2.3.9)

Column-prefix similarities are added to the similarity score, when doing an in-row substitution movement, and row prefix similarities are added, when doing an in-column

substitution movement. In the case of the combined diagonal movement, the order then impacts which row prefix and which column prefix is considered.

Let the upper index in $S^{-i,-j}$ denote which array prefix is used: $S^{-1,0}$ is S without the last row, $S^{0,-1}$ is S without the last (rightmost) column, and $S^{-1,-1}$ is S without both last row and column. indel(S) is the indel score of the bottom right element of S.

To formalize (compare with equation 2.3.2), the similarity of two arrays S_1 and S_2 is defined as:

$$\text{sim}(S_1, S_2) := \max \left(\begin{array}{cc} 0 & \\ \text{sim}(S_1^{-1,-1}, S_2^{-1,-1}) + \text{sim}(S_1 \uparrow, S_2 \uparrow) + \text{sim}(S_1^{0,-1} \leftarrow, S_2^{0,-1} \leftarrow) & \text{(I)} \\ \text{sim}(S_1^{-1,-1}, S_2^{-1,-1}) + \text{sim}(S_1^{-1,0} \uparrow, S_2^{-1,0} \uparrow) + \text{sim}(S_1 \leftarrow, S_2 \leftarrow) & \text{(II)} \\ \text{sim}(S_1^{0,-1}, S_2^{0,-1}) + \text{sim}(S_1 \uparrow, S_2 \uparrow) & \text{(III)} \\ \text{sim}(S_1^{-1,0}, S_2^{-1,0}) + \text{sim}(S_1 \leftarrow, S_2 \leftarrow) & \text{(IV)} \\ \text{sim}(S_1, S_2^{-1,0}) + \text{indel}(S_2) & \text{(V)} \\ \text{sim}(S_1, S_2^{0,-1}) + \text{indel}(S_2) & \text{(VI)} \\ \text{sim}(S_1^{-1,0}, S_2) + \text{indel}(S_1) & \text{(VII)} \\ \text{sim}(S_1^{0,-1}, S_2) + \text{indel}(S_1) & \text{(VIII)} \end{array} \right) \tag{2.3.3}$$

This similarity is then implemented in the same way as the standard Smith-Waterman approach, with time complexity in $O(N \times M)$, where N is the number of elements in S_1 and M is the number of elements in S_2. Although this algorithm is generally designed to detect patterns in arrays, the actual use in Lecroq et al. [2012] is to identify similar passages of annotated conversations. This kind of data is notably sequential only in the time dimension, whereas the annotation dimension is fixed, and each column has its own alphabet. Therefore, the application is to a problem of finding similar subsequences in sequences of tuples.

With these different approaches having been exposed, we can now compare their suitability for the routine activity detection problem.

2.3.2.3 Discussion

Our look at string mining is focused on alignment techniques, as this class of approaches returns pairs of similar subsequences from two input sequences of symbols and thus allows us to *extract* information from the data. There are three identifiable ways how to apply this to our routine activity detection problem:

- reduce the sensor data to discrete context states, and obtain a single sequence of such states – each similar pair of subsequences should correspond to a routine activity;

- formulate the task as a multi-sequence task (to remain coherent with multiple sources of context data) and seek alignments on each sequence – merged similar subsequences should correspond to routine activities;

or

- consider context to be a sequence of n-tuples, with each tuple corresponding to a context state, consisting of multiple discrete context factors – pairwise locally similar subsets should correspond to routine activities.

These three different conceptual approaches are illustrated in Fig. 2.3.10.

In each case, it is required that the context is represented in a discrete format. The difference between the first and following two approaches lies in the alphabet size and tolerance to desynchronisation. A single value that encodes multiple values cannot encode certain intricacies in context data, such as one sensor reading leading or lagging the same sensor reading in another instance (5), with regard to the other sensor data. This restricts the appeal of the single string approach.

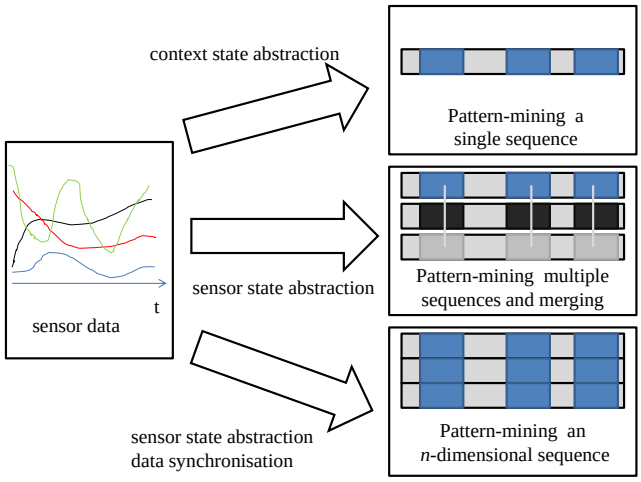

Figure 2.3.10.: Three different ways of modelling context for different sequence mining approaches. "Data synchronisation" refers to the need to have a full set of sensor readings available at every time step, irrespective of different sampling frequencies of the actual sensors.

The multi-sequence approach is excellent with regard to desynchronisations, as each sensor is studied at an independent temporal time scale from the others, to find patterns. On the other hand, this complete uncoupling means that coupling effects only get introduced after the first pattern mining pass, introducing additional model parameters (6) to characterise the merge operations.

The multi-dimensional approach is limited, in that there are no efficient ways of mining such patterns. On the other hand – as the illustration makes readily apparent – it is possible to project the data into the plane, which induces "neighbourhood artefacts" where the link between neighbouring sensors is stronger than between those that are projected into areas that are further apart. The Smith-Waterman based two-dimensional local alignment algorithm can find patterns in this data, but at the cost of relatively high complexity (1).

The exact motif-extraction approach of Apostolico et al. [2008] is also capable of extracting information from such projected context data, but the time complexity of its combinatorial approach is even higher (1). Additionally, it will only extract exact patterns, having only the freedom of the *don't care* symbol, but not permitting structural variation between pattern instances (5). Although the exact approach means that initial parametrisation is not required, by consequence it is not possible to obtain an intrinsic distance between two patterns, or to allow for substituted elements outside of replacements with the *don't care* symbol, which removes all information of the subset of symbols that can appear in those positions. With sufficient pre-treatment of the context data, in theory it could be possible to reliably extract patterns, of flexible shapes. In comparison to the alignment approach the complexity is too limiting, and the amount of pre-treatment too vast, for this approach to be considered suitable for the routine activity detection problem.

Similarly, although the suffix-tree solution to the dispersed repeat problem has efficient solutions (1,7), the fact that it merely covers exact repeats, means that the pre-treatment has to deliver a sequence of activities as input (2,3). Otherwise the variations which one can expect in context data, render the approach unsuitable (4,5). Such a pre-treatment is difficult to obtain in a sufficiently general manner, which renders exact repeat solutions more difficult to apply to the routine activity detection problem than alignment solutions.

2.3.3 Comparison

For each of the two approaches, we have presented similar classes of approaches. Looking back at the list of seven criteria we initially defined, we can rank features of classes of approaches for each one:

Complexity (1) The first criterion favours approaches which have low algorithmic complexity and also an inherent capacity to deal with multidimensional data. Whereas the former is an obvious implication, the latter judgement is based on the fact that any other approach would either require a merging post-processing step or projecting pre-processing, both of which may have unexpected implications on complexity.

Preprocessing (2) The second criterion similarly favours multi-dimensional approaches, as pre-treatment can be reduced to the per-element level and possibly a simple synchronisation.

Adaptability (3) The third criterion is somewhat in opposition with criteria (2) and (6). Complex models are able to closer match the actual data, and give a richer representation. Approaches using multi-level hierarchical or substitution score based models have advantages under this criterion.

Noise (4) Extraction from noisy data is best performed by approximate approaches. Exact approaches require additional pre-treatment to fulfil this criterion.

Time Effects (5) This criterion also favours approximate approaches, specifically those that allow skipping or ignoring individual entries when mining for patterns, as well as treating each sensor with a certain amount of individuality.

Parametrisation Effort (6) The parametrisation effort is lower for substitution based models compared to hierarchical models, as there is no need to define cross-sensor relations.

Partial Mobile Deployment (7) This final criterion mostly an engineering challenge, as all sequence mining based approaches require a large off-line component, that does not need to be based on the mobile device. On the other hand, a recognition or prediction algorithm based on an established model can be deployed on a mobile device with little worry about platform constraints.

For both string mining and SPM, we can at this stage discard the exact one-dimensional approaches – exact repeat mining and the standard sequential pattern mining algorithms – as the demands this would put on the pre-treatment of the data cannot reasonably expected to be met.

Table 2.1.: Comparison of SPM and string mining

	sequential pattern mining				string mining			
approximative	Y		N		Y		N	
multi-dimensional	Y	N	Y	N	Y[2]	N	Y	N
(1) complexity	X	high	med.	low	high	med.	high	low
(2) preprocessing		high	med.	high	low	high	med.	high
(3) adaptability		med.	med.	low	high	med.	low	low
(4) noise tolerance		high	med.	low	high	med.	med.	low
(5) time effects tolerance		low	med.	low	high	low	low	low
(6) parametrisation effort		med.	high	low	high	med.	low	low
(7) mobile deployment		no	no	yes	no	yes	no	yes

The approximate methods of string mining have a larger maturity compared to the approximate approaches of sequential pattern mining, or in the case of the approach by Kum et al. [2003] is even based on a string mining paradigm. Additionally, there remains a drawback of sequential pattern mining, in that there is no consecutivity criterion present in most algorithms.

Plantevit et al. [2010]'s multi-dimensional approach based on the M^{3}SP algorithm is the only true multi-dimensional algorithm in both fields. To some degree, it shares the drawbacks of the exact sequential pattern mining approaches, but due to the multidimensionality, the pre-treatment of the data becomes much less of an issue, and the underlying hierarchical model can be used to give a semblance of an approximate approach. A way to render it more suitable to the routine activity detection, would be to add a consecutivity criterion, as has been done with the algorithm of Pinto et al. [2001] by Rashad et al. [2007a] or take into account temporal data [Giannotti et al., 2006, Yoshida et al., 2000]. The two-dimensional approach by Lecroq et al. [2012] appears to be similarly suitable. The key drawback lies in the required projection of the multi-dimensional context data into the plane in which the algorithm operates, but it takes into account desynchronisations across different sensors, and could be reduced to a lower complexity by taking into account the projection, and thus the incompatibility of data that is adjacent in the sensor-dimension. A side-by-side comparison of the general approaches with regard to the seven criteria is presented in Table 2.1.

[2]This is an estimate based on a naive extension of the approach of [Lecroq et al., 2012] to multiple dimensions.

2.4 Conclusion and Direction

Our survey of available context datasets has shown that none of them is truly suitable to quantifiably verify that detected patterns correspond to routine activity. The lack of ground truth is the prime limitation, but there is also only a small subset of physical context data present in the existing datasets. This leads us to pursue two strategies to address – separately – each of the shortcomings: A generation algorithm can give us true ground truth knowledge, whereas our own data collection campaign would provide us with as much physical information as required. We also need to annotate some of the real world data with routine activities, to quantitatively verify our claims.

With regard to the algorithms and approaches we studied, none matches our task directly. All require either modification or extensive adaptation to a specific dataset. A common drawback is the substantial effort required to prepare context data for the data model used in conjunction with the algorithms; alignment based approaches minimise this aspect, by only requiring tables of similarity scores, which can be determined using statistical analysis. Alignment approaches are also uniquely tolerant to desynchronisations between sensor streams. In the above comparison to SPM, the alignment approaches win out, barely. Compared to a clustering or a probabilistic state machine approach, the simpler pre-processing of the alignment approach is the deciding factor in its favour.

This leads us to pursue an approach based on n-tuple sequence alignment.

Alignment of Sequences of n-Tuples

Overview

This chapter presents our algorithmic contributions towards an answer to the routine activity detection problem. First we introduce a model that presents context in a way that an alignment algorithm can use. Next we cover our algorithmic contributions to the alignment algorithms of the state of the art, and finally we discuss the implications of our approach.

3.1 Context Model

Fundamentally, context data is the set of values of a number of context factors, at a specific moment in time. Assuming there are n context factors, this is most accurately reflected as a vector-valued function over time, of the type $C(t) = (c_1(t), c_2(t), \ldots, c_n(t))$. Of course, a digital computer can impossibly treat continuous data of any kind, and a digital sensor can never capture data in a continuous manner. Any context *data* therefore is a discrete valued time series, unless different context factors are sampled at different rates, in which case the data is in the form of multiple time series. We adhere closely to this natural representation in the following.

3.1.1 Sequence of n-tuples

We postulated in the introduction that routine activities correspond to recurrent motives in context data. The state of the art approaches to finding such motives favour the use of a sequence based approach. Consequently, we adopt a model that represents context as a sequence of context states. In specific, we structure context as a sequence of n-tuples.

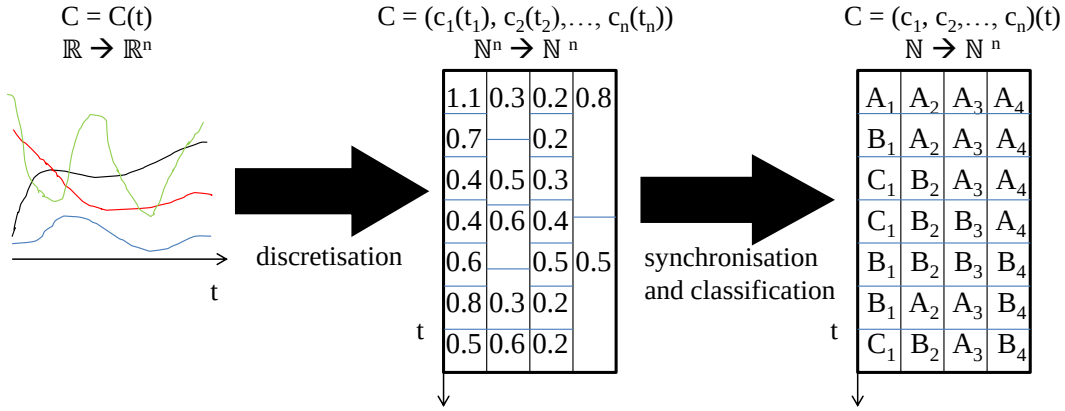

Figure 3.1.1.: The transformation of continuous context $C(t)$ into context data in a sequential model.

With regard to the natural time series representation, this requires that our context data sources are synchronized and sampled at constant and equal intervals. Each position in a tuple contains a (coarsely – e.g. at a semantically relevant level) discretised or classified reading from one of n context data sources. These context data sources each measure one context factor and are referred to as sensors in the following. Each tuple represents context at a moment in time, which we assume to be valid over the constant interval of time, until the next set of values becomes available.

In Fig. 3.1.1 the process of transformation from physical context into context data and finally into a synchronized and coarsely discretised n-tuple representation is illustrated. Discretisation (at the physical digital sensor level) transforms context from the continuous function over time into a number of time series. This is then synchronised into a single time series, and the values are replaced with abstract representative class IDs, column by column. In the example, we excluded possible expansions or contractions in the "width", due to a single sensor measuring multiple physical properties, or data from multiple sensor being used to determine a class based on multidimensional data. This is merely to aid comprehension — in reality these characteristics may appear in the process chain.

By choosing an n-tuple approach to alignment, over a 2D-approach, we can eliminate a degree of freedom from the latter. The fact that each sensor has its own, separate set of values, allows us to fuse the two in-tuple dimensions into a single one, by requiring that any operation performed along one axis has to be equally performed along the other. This enforced translational (across sequences) symmetry not only prevents us from having to define similarity values for nonsensical configurations (e.g.: How similar

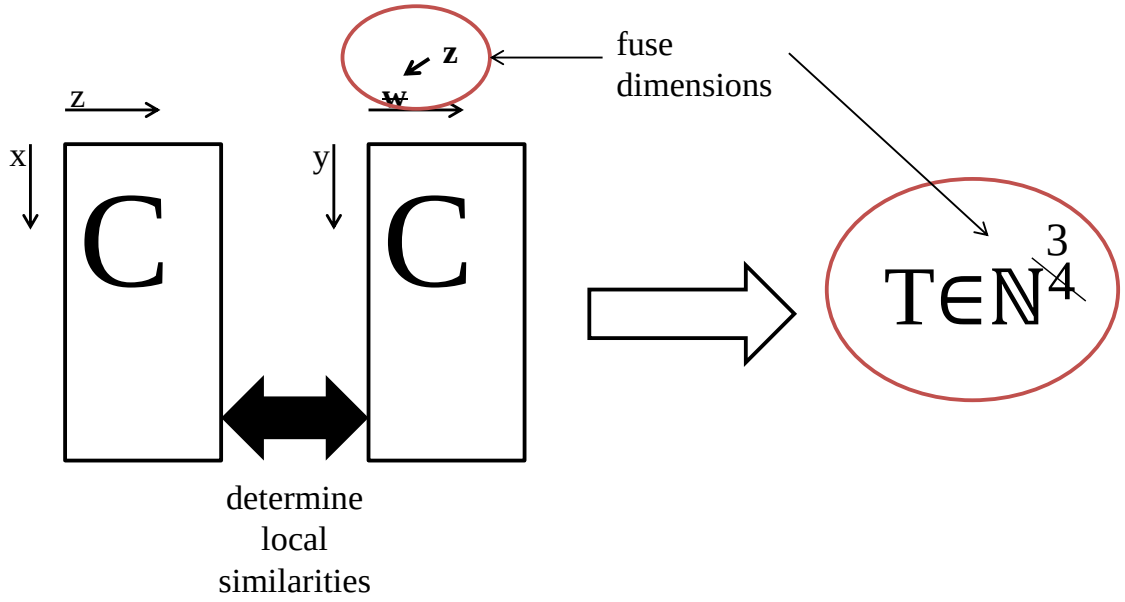

Figure 3.1.2.: Linking w- and z-axes into a single dimension, due to translational symmetry (i.e.: every set of discrete steps taken along the z-axis is replicated identically in the w-axis).

is a bright light to an upside down mobile device?) but also reduces complexity of the alignment algorithm. The specific consequences are discussed in subsection 3.2.1. Fig. 3.1.2 shows how the four-dimensional alignment problem has been reduced to a three-dimensional one, by linking the in-tuple dimensions.

3.1.2 Blocking and Sampling

Although it is generally possible to find pairs of similar subsequences by searching for local alignments of a sequence with itself, the time and memory required to do so grow by the square of the length of the sequence (cf. paragraph Local Alignment on page 35). There exists a limit from which on it is no longer possible to calculate local alignments, because no computer system has sufficient memory available to perform the computation.

We have identified two means of managing the issues that arise from an increased volume of data:

- A first variable that we can control a priori is the sampling frequency of context data. Depending on the granularity of context patterns we wish to identify, we can reduce or increase the sampling frequency. This also reduces or increases the sequence length, for a given time interval of context data.

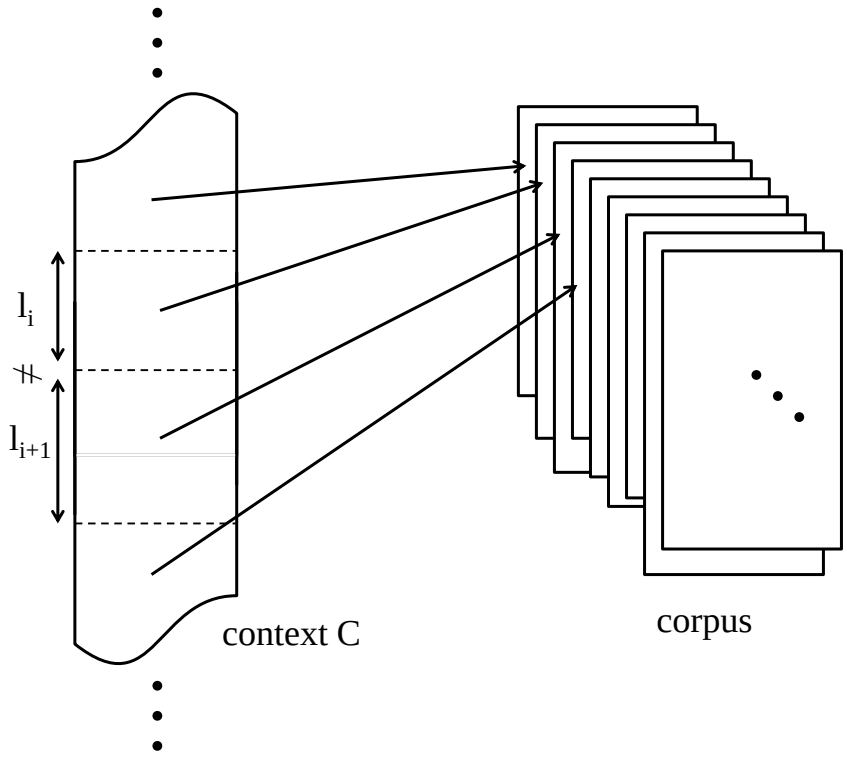

Figure 3.1.3.: Splitting a long sequence of context data into blocks of roughly equal (but not necessarily equal) lengths.

- Another way to address this problem is by splitting the sequence along regular (time-)intervals into subsequences of n-tuples. It is often sensible to do this along semantically relevant lines, for example by choosing 24 hour intervals or seven day intervals. This is illustrated in Fig. 3.1.3 where a long sequence of context data is divided into a set of subsequences with potentially different lengths. The different lengths are artefacts of the data collection, and do not affect the alignment calculation.

Memory constrained situations are defused by using this approach, as the accumulated score tables for each pair of subsequences are much smaller. Fig. 3.1.4 shows how the large table spanned between two long sequences a and b is reduced to many smaller tables, when the long sequences are divided into blocks. The impact on total computation time is negligible, due to high number of alignment operations that result, which also scales to the square of the number of segments. On the other hand, it is trivial to parallelise across pairs of blocks, which reduces the real duration (i.e. wall time) of performing an alignment when more than one processor is available.

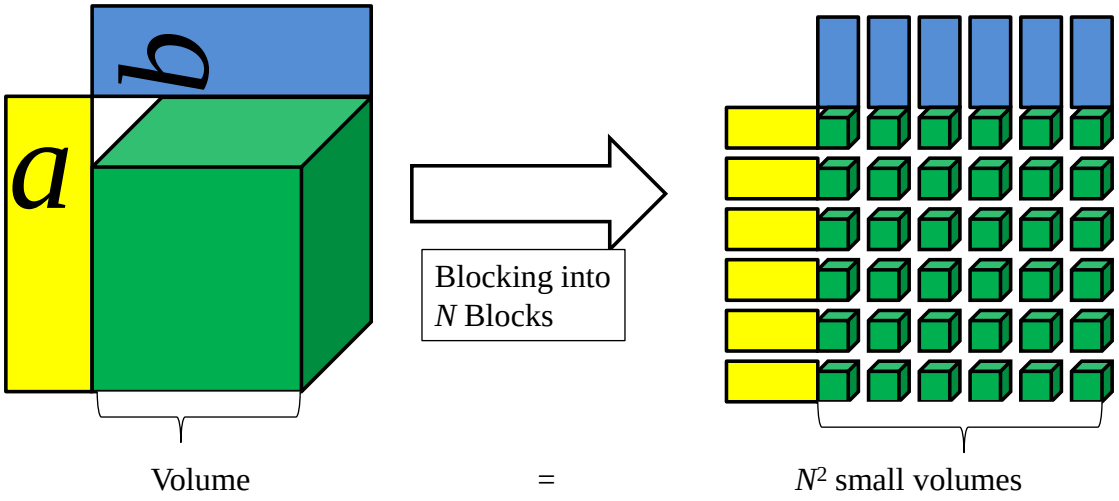

Figure 3.1.4.: The reduction of the size of individual accumulated similarity score tables which can be obtained by blocking. a and b are two sequences, with the local similarity scores contained in the volume spanned between them.

Another advantage to blocking, lies in that fact that blocks are crucial for a sequence mining approach. Conversely, a downside is that there is an accuracy penalty, even when choosing "convenient" intervals (i.e. such that periods of interest are well away from the beginning or end of a block). Notably, patterns that fall across these boundaries could be represented as two separate alignments, or not at all, because each half might be too short to meet an imposed similarity criterion. One possible means of addressing this shortcoming, is to allow overlaps between adjacent blocks. This practice introduces some overhead, when alignments that can be found in the overlap are "stitched together".

3.1.3 Meta-data

As we pursue a local alignment-based approach, we need to be able to determine local similarities of subsequences. A key requirement is that we define the relation between each pair of discrete states of each sensor by means of a set of similarity scores. These similarity scores are by nature positive, when describing the similarity of identical values, and negative when describing the similarity of non-identical values. We chose to represent these values in the form of n (one for each element of an n-tuple) symmetrical tables, containing positive values in the main diagonal, and negative values elsewhere. A small example is given in Fig. 3.1.5.

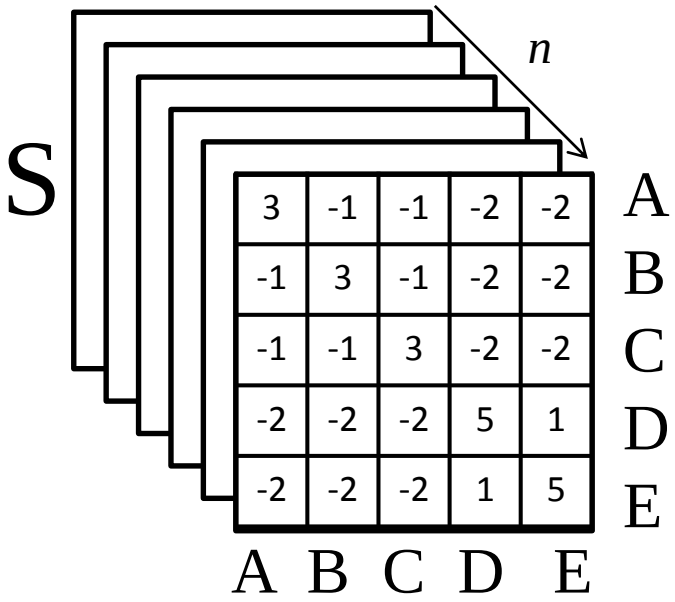

Figure 3.1.5.: A sample substitution similarity score table from the set of n tables.

We can imagine four ways, how such a set of meta-data can be obtained. The first two are based on statistical analysis of an existing dataset: transition frequencies between sensor states are a possible indicator to an underlying system, but the same can be claimed of substitution probabilities between sensor states for hand-selected patterns. A third approach is to base the values on the physical distances of the underlying classes. In the case of two places, the distance of the shortest route between the two, or the time required to cover that distance, could be such physical pointers. Finally, in absence of such data, a simple model that does not assign different scores at all, except one positive and one negative score to differentiate between same and different values can be used. Each approach requires expert supervision, and of course it is also possible for an expert to project his own view of the problem onto a manually crafted set of meta data that does not directly reference any of the above approaches, or mixes them.

Besides these substitution scores, an alignment-oriented model also requires insertion and deletion scores. Contrary to the affine (for length) and constant (for deleted/inserted value) approach chosen in bioinformatics (cf. BLOSUM-type block transition score tables by Henikoff and Henikoff [1992] and the work of Altschul and Erickson [1986]), we decide to use scores which are a fixed offset of the substitution similarity score.

Our reasoning behind this choice is as follows: in context data it is common for context sources to return a constant value, for a different length of time. Particularly, periods of inactivity (with regard to the mobile device) show this characteristic. An example:

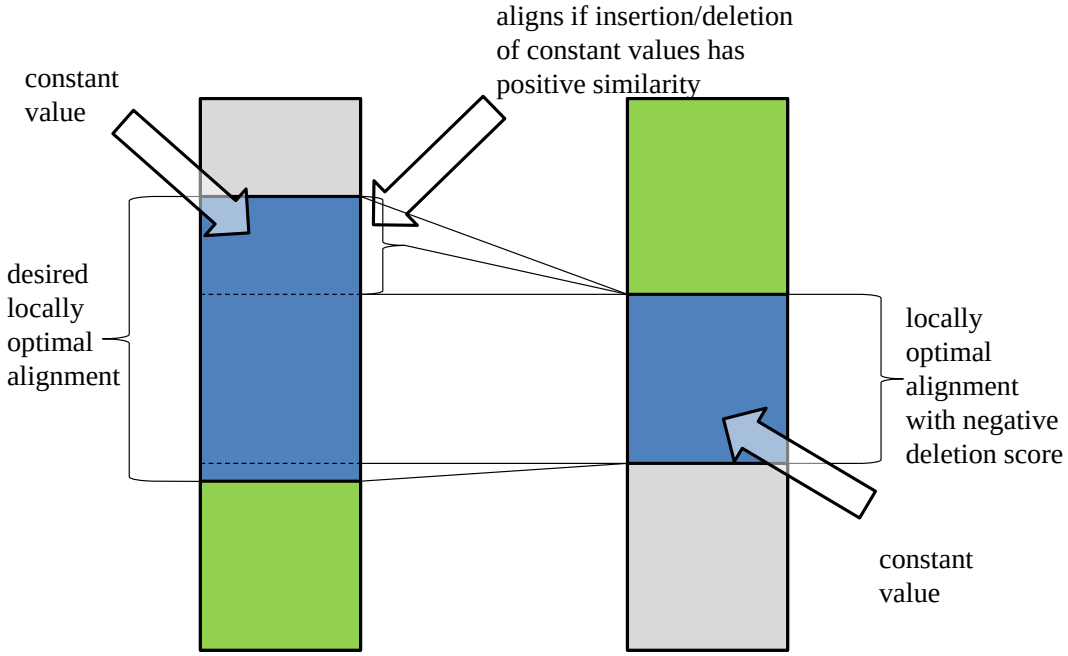

Figure 3.1.6.: Two intervals of identical values but different lengths can be aligned by assigning positive similarity score to insertions and deletions of identical values.

a user sleeps for different lengths of time during two different nights in a week. Our goal in this case is to align the entirety of both instances of "user sleeps", instead of just the closest matching sub-sequence of the longer interval. The score offset gives a high similarity score for alignments of a subsequence of constant values with another subsequence of identical values but different length, while also penalising non-identical deletions and insertions over substitutions.

This concept is illustrated in Fig. 3.1.6, where two intervals of constant values but different lengths are shown side by side, with two possible ways of defining similarity being shown: To the right, the state-of-the-art approach of assigning negative scores to all deletions and insertions, and to the left an example where the sum of deletion score and substitution score of the two constant values is positive. This allows the inclusion of the entire interval. In practice, a positive score would usually be undesirable, as problems of scale might arise. Often it is sufficient to allow a similarity score to bridge across such intervals without penalizing the accumulated score too much, as activities following longer series of constant values are usually also similar, if the constant values are part of the trace of a significant routine activity. In this case a lightly negative score-sum is preferred.

3.1.4 Discussion

This model – and particularly the definition of a similarity measure – is specific to alignment approaches. Data that is presented according to this model can be transformed to conform to another model, as long as the granularity of the discretisation is not too coarse, and the classification key which is used to map raw values to abstract context values is available. Of specific interest to us, is the transformation to a model that is compatible to a sequential pattern mining approach, because it would enable a direct comparison. Such a conversion would require that the similarity tables are transformed into a similarity hierarchy.

We consider the absence of semantic and physical information in data that conforms to this model to be an advantage: it protects the privacy of the user whose data is being treated. Despite it being possible to infer some semantic information (e.g. which location IDs correspond to home and work) from a stream of data formatted according to this model, there is very little risk of physical information (i.e. where the previously mentioned places are located) to become compromised. This makes our model suitable for data storage and processing on distributed systems, that are not necessarily under direct control of the user, with little risk of a breach of privacy. This hypothesis is reinforced by the evaluation of Voigtmann et al. [2012] of different context analysis approaches and models.

The adaptation of raw data to our model requires some amount of intervention by an expert, but automation is possible to a degree. The influence of this expert in the creation of the model is pivotal. Assuming that an expert has derived a perfect context model, we cannot guarantee that he can transform this model with perfect accuracy into a set of meta-data. This limitation is due to the fact that similarity values are limited in precision and difficult to scale across multiple sensors.

Anecdotally, in the field of biological sequence alignment, research showed that incorrectly obtained substitution score tables (Styczynski et al. [2008]) can in fact increase the accuracy of the used alignment algorithm. In this case an error in the statistical determination of similarity values from a reference dataset was present. We therefore theorise that our similar model would display similar resilience to slight inaccuracies in the similarity value tables. We also have to emphasise that the parametrisation of a model that performs exactly as expected is far from a well understood or intuitive practice.

With this model in place, we examine in more detail our approach to process this data for routine context.

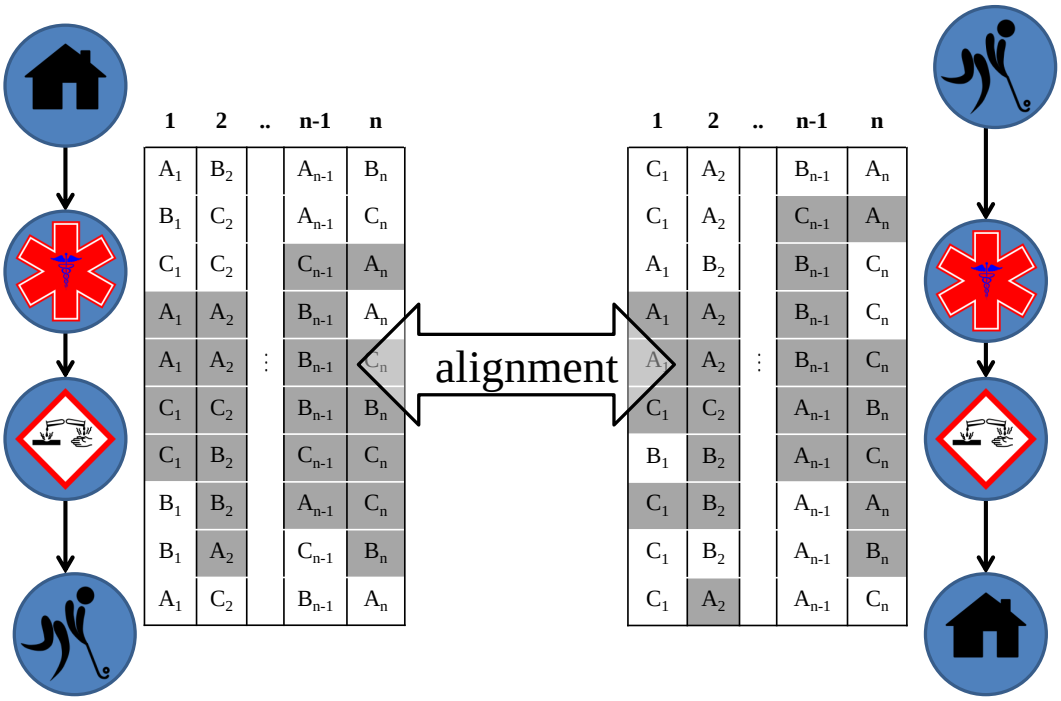

Figure 3.2.1.: Two example context sequences, the resulting context data sequences and an alignment (cells underlaid in grey) that corresponds to two similar subsequences.

3.2 Alignment Algorithm

It is our declared goal to find patterns in context data. As we have laid out in our survey of the state of the art (cf. section 2.4 on page 46), the search for local alignments appears to be the most suitable approach to this task. Any such alignment (i.e. a pair of similar subsequences of two sequences from a corpus of context data) corresponds to two instances of similar activities or contexts. Fig. 3.2.1 reprises how context sequences correspond to alignments. Two similar sequences of activities (being at home, working at the hospital in the morning and the laboratory in the afternoon, before playing hockey on one hand and the same activities in a different order on the other) result in two similar sets of context data. A subset of this data attains a locally optimal similarity score and is therefore considered to be in alignment. This alignment then serves as pointer for the original similarity in the activities.

Fig. 3.2.2 illustrates how pairs of context sequences from a corpus (i.e. an established set of subsequences of context data - cf. subsection 3.1.2 on page 49) are generally aligned. This is the key operation to find frequently appearing similar subsequences in

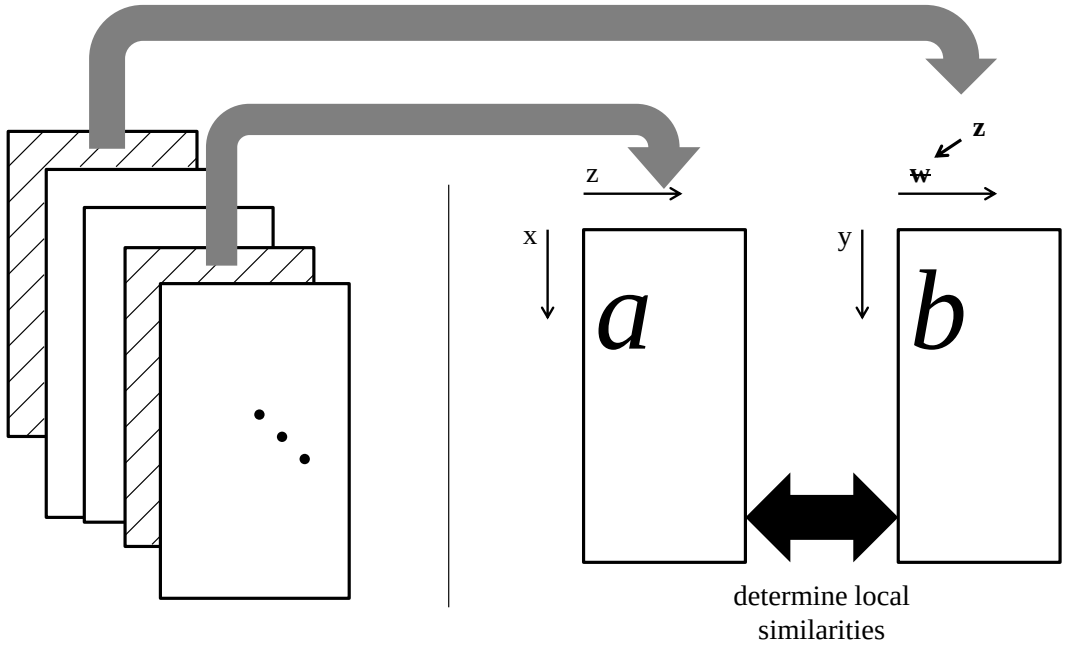

Figure 3.2.2.: Alignment of two context sequences a and b from a corpus.

this corpus, which we then understand to be representatives of routine activities.

Before we continue, we briefly discuss the two key terms that we use so frequently in this work: *alignment* and *similarity*. An alignment is defined as a result of an alignment algorithm with a specific configuration. More generally, it is a subsequence and the associated operations to transform it into another *similar* subsequence (cf. subsection 2.3.2 on page 25).

On the other hand, the – rather abstract – notion of similarity is less obvious. With regard to sequences, we can define four key criteria of similarity: Substitution similarity at the elemental level is the most atomic criterion. It stems from the pair-wise similarity values defined beforehand (cf. subsection 3.1.3 on page 51). Another similarity criterion is the density of positive substitution similarities. A high density means a high similarity of a set of values. A third similarity criterion is size: at the same density, a larger set of values can be considered to be more similar, than a smaller one. Finally, we impose a synchronicity criterion. The less gaps need to be opened or filled in a pair of subsequences, given the same size and density, the higher the similarity score for the overall sequence.

In the following, we implement this notion of similarity with a recursively accumulated similarity measure. Basically, we call a sequence "similar" to another, if the last row and column of both sequences are similar, and the remaining sequence is also similar

(cf. subsection 2.3.2 on page 25 and the next subsection for formal descriptions), while using a standard one-dimensional similarity measure for rows and columns. This measure allows us to describe a very precise notion of what similarity is. By parametrising the context model appropriately, we can craft a measure that corresponds to specific expectations. An expert – who defines a set of rules that effectively links certain kinds of patterns in the data to routine activities – can therefore tune the parameters on a case-by-case basis.

A similarity measure based on this concept lies at the heart of the work of Lecroq et al. [2012] on aligning annotated dialogues to find similar structures. We extend and adapt this approach for use on context data and our context model. Our modifications to their algorithm, which primarily address significant performance issues that arise when attempting to use their approach on long sequences of context data, are detailed in the following.

3.2.1 Contribution I: Reduction to n-tuple Problem

Our first angle of approach lies in the elimination of unnecessary size of the table of accumulated scores. In the original approach, this table spans four dimensions, to take into account insertions and deletions in horizontal and vertical directions in the plane. As our context model does not allow for any interaction between different elements of the same tuple, we restrict permitted operations from the 2D approach (cf. subsection 3.1.1 on page 47 and Fig. 3.2.2 on the preceding page).

When determining the similarity of a pair of tuples we now only allow substitutions between elements with identical indices. This brings our approach closer to a true n-tuple approach. As a consequence, insertion and deletion operations are restricted to the sequence dimension, but are *not* limited to entire n-tuples at a time. This optimisation reduces the local similarity score table T to three dimensions, and thereby each cell only depends upon five "predecessor cells". Each of these cells corresponds to a vertex in the cube marked T in Fig. 3.2.7.

This step also simplifies and thereby optimises some ancillary calculations. It is often required to calculate row and column similarity scores locally; for the row case, only substitutions are of interest. The local column alignments also mean that the algorithm retains full flexibility when working with data that has a tendency to de-synchronise, by being able to "break up" tuples to form alignments.

As the calculation of the score table is naturally defined in a recursive manner, we will first define ways of selecting individual tuple elements in a sequence with a reference point in the bottom right. For the following, let s be a sequence of n-tuples of length l.

Definition 1. $s\,[i,j]\,,\ i < l \in \mathbb{N}, j < n \in \mathbb{N}$ is the element in the j-th position from the end of the n-tuple in the i-th position from the rear (l-th row) of the sequence s.

In Fig. 3.2.3, we present three examples (one general, and two similar to the common usage in the following) of how this inverted addressing of elements of a sequence works.

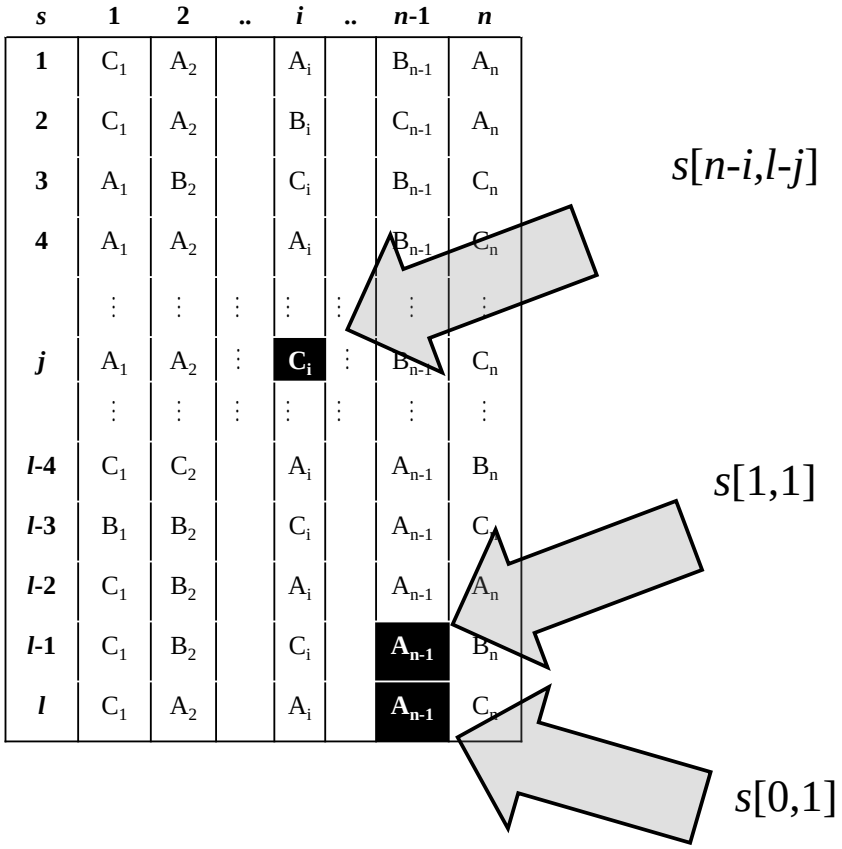

Figure 3.2.3.: Selection of tuple elements in a sequence using reverse addressing. The origin of the coordinate system used here is in the lower right end of the sequence. From there the first coordinate is incremented when moving one element to left, and the second when moving one element upwards.

Next, we define a way to express the recursive reduction of the sequence:

Definition 2. The $\bullet_{i,,j}$ operator represents the sub-sequence consisting of the original sequence, minus the last i tuples and the last j tuple elements. If we understand s to be an ordered set of coordinate-value pairs, this can be expressed as $s_{i,,j} := s \setminus s\,[x,y]\,\forall x < i \vee y < j$.

This operator is illustrated in Fig. 3.2.4, where both a general case is shown, and the

usage that is most common in the following, with indices in the [0,1] range.

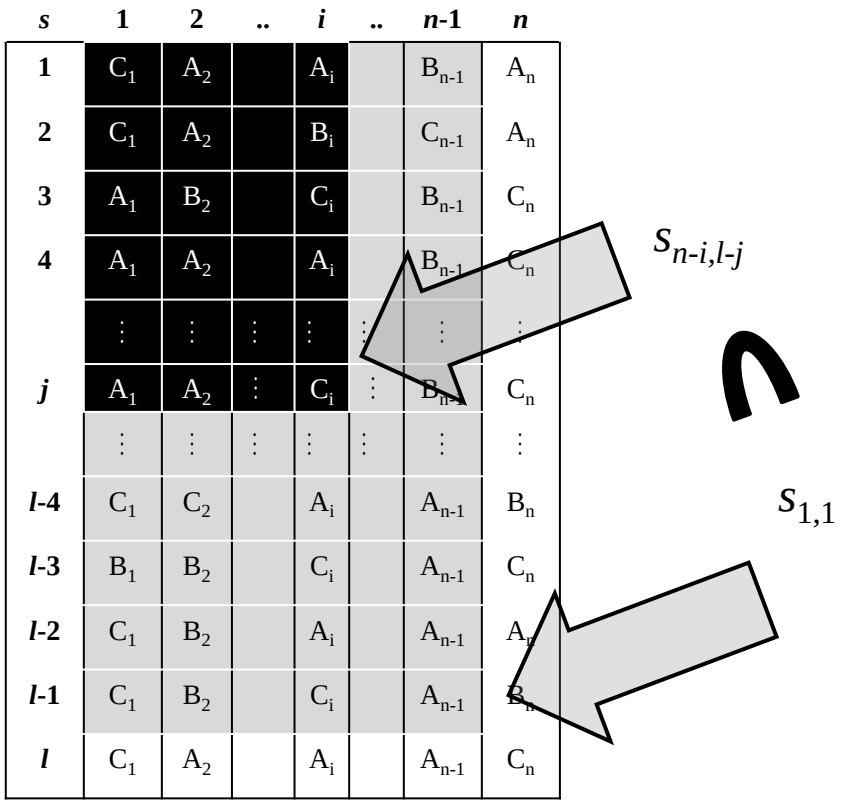

Figure 3.2.4.: Example of the $\bullet_{i,,j}$ operator. If the operator is used on the same sequence of n-tuples, with one pair of indices being higher than the other, the smaller result is a subset of the larger one. In this case the result underlaid in black is a subset of the result underlaid in grey.

To calculate in-tuple and in-column similarity values, we define two operators:

Definition 3.

$$s\left[i,j\right]\leftarrow:=\left\{s\left[i,j+1\right],s\left[i,j+2\right],\ldots,s\left[i,n\right]\right\}$$

is the right-to-left sequence of elements to the left of a position in a tuple, and

$$s\left[i,j\right]\uparrow:=\left\{s\left[i+1,j\right],s\left[i+2,j\right],\ldots,s\left[l,j\right]\right\}$$

is the bottom-to-top sequence of all prior elements in a column (i.e. all elements with the same tuple index).

One example of each of these operations is illustrated in Fig. 3.2.5.

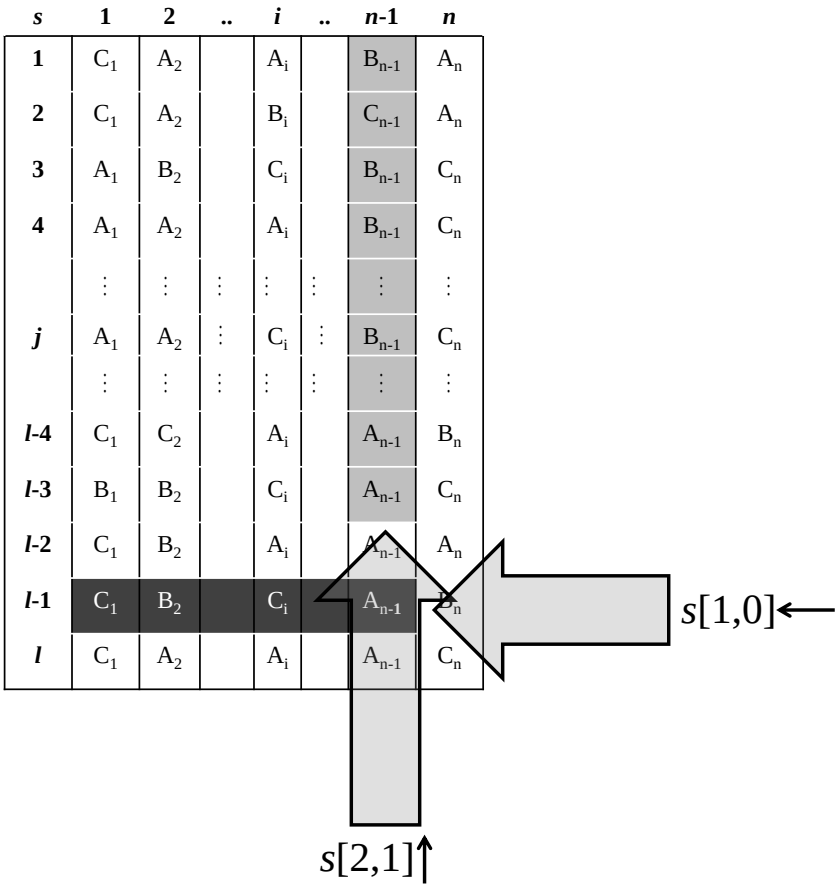

Figure 3.2.5.: Selection of a column and row using the ↑ and ← operators.

Lastly, we require a uni-dimensional similarity measure, to determine what used to be row and column scores:

Definition 4. For two sequences x, y of single elements, $\mathrm{sim}(x, y)$ is the

- locally (i.e. before the similarity reaches a zero value) maximum similarity score of the Smith-Waterman algorithm on the reversed column sequences, if x and y are both sequences generated with the ↑ operator.
- maximum accumulated value of substitution similarity scores, if x and y are sequences generated by the ← operator.

Fig. 3.2.6 shows an example to illustrate this, based on 4 steps: First a pair of columns is extracted from a sequence of n-tuples, then reversely aligned, one with the other. Once the similarity score reaches zero, the alignment is aborted, and the local maximum in the aligned interval is the score returned by the sim operator.

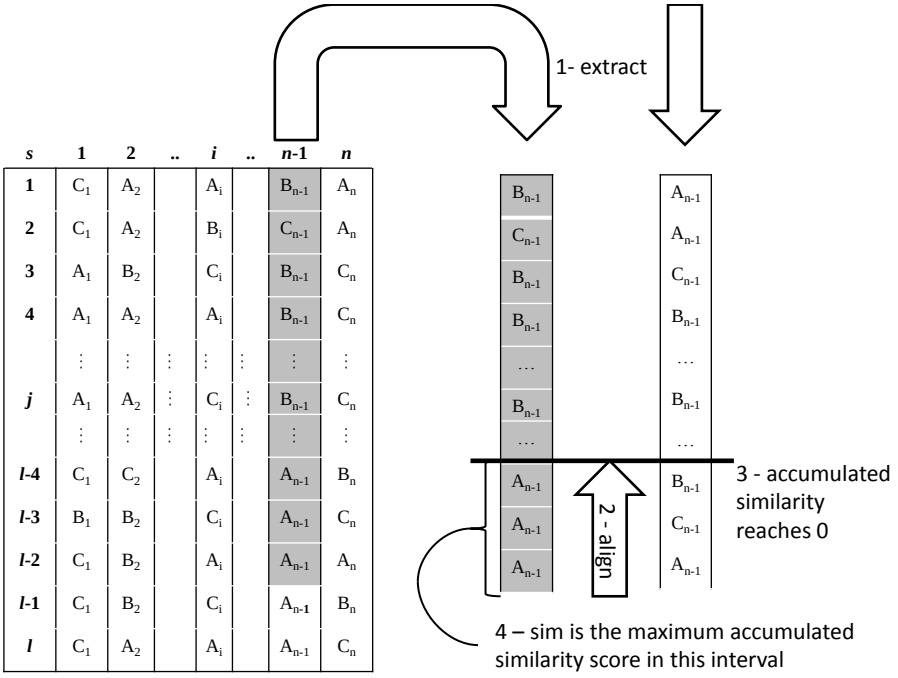

Figure 3.2.6.: Calculation of a column similarity score. Two columns are extracted from two sequences, then aligned in inverse order, up to the point where the similarity score reaches zero. The maximum similarity from this interval is the similarity value we assign these two columns.

With these prerequisites in place, we can express the accumulated similarity score for two subsequences a and b recursively as follows:

Definition 5. For two sequences of n-tuples a, b and the accumulated similarity score is:

$$\mathrm{sim}(a,b) = \max \left(\begin{array}{cc} 0 & \\ \mathrm{sim}(a_{1,1},b_{1,1}) + \mathrm{sim}(a\,[0,0]\uparrow, b\,[0,0]\uparrow) + \mathrm{sim}(a\,[0,1]\leftarrow, b\,[0,1]\leftarrow) & \text{(I)} \\ \mathrm{sim}(a_{1,1},b_{1,1}) + \mathrm{sim}(a\,[1,0]\uparrow, b\,[1,0]\uparrow) + \mathrm{sim}(a\,[0,0]\leftarrow, b\,[0,0]\leftarrow) & \text{(II)} \\ \mathrm{sim}(a_{0,1},b_{0,1}) + \mathrm{sim}(a\,[0,0]\uparrow, b\,[0,0]\uparrow) & \text{(III)} \\ \mathrm{sim}(a_{1,0},b_{1,0}) + \mathrm{sim}(a\,[0,0]\leftarrow, b\,[0,0]\leftarrow) & \text{(IV)} \\ \mathrm{sim}(a_{0,0},b_{1,0}) + \mathrm{indel}(b[0,0]) + \mathrm{sim}(a[0,0]\leftarrow, b[1,0]\leftarrow) & \text{(V)} \\ \mathrm{sim}(a_{1,0},b_{0,0}) + \mathrm{indel}(b[0,0]) + \mathrm{sim}(a[1,0]\leftarrow, b[0,0]\leftarrow) & \text{(VI)} \end{array} \right) \tag{3.2.1}$$

The similarity score $\mathrm{sim}(a,b)$ is 0, if $a = \emptyset \vee b = \emptyset$.

In the domain of the accumulated local similarity score table T, each cell contains the similarity values of the subsequences (of sub-tuples) defined by the coordinates (cf. subsection 2.3.2). The iterative algorithm of determining each value in T is started by initialising the first plane in each dimension of the table with zeroes, to satisfy the end condition of the recursive definition of the similarity scoring function. The other cells are calculated as illustrated in Fig. 3.2.7: Each arrow (labelled I-VI) corresponds to an operation in equation 3.2.1, excluding the column and row similarities. These are the operations that are performed in each case, before choosing the maximum value amongst them:

(I) Substitution and movement to the left (column-row-order): The space diagonal predecessor value is added to the column similarity of the predecessor and the row similarity of the current rows.

(II) Substitution and movement to the left (row-column-order): The space diagonal predecessor value is added to the column similarity of the current columns and the row similarity of the predecessor.

(III) Movement to the left: The z-axis predecessor value is added to the column similarity of the current columns.

(IV) Substitution: The x-y-diagonal predecessor value is added to the row similarity of the current rows.

(V) Insertion: The y-axis predecessor value is added to the row similarity of the current row and preceding row.

(VI) Deletion: The x-axis predecessor value is added to the row similarity of the preceding row and current row.

The dynamic programming paradigm of the approach is therefore maintained, despite our recursive definition of local similarity.

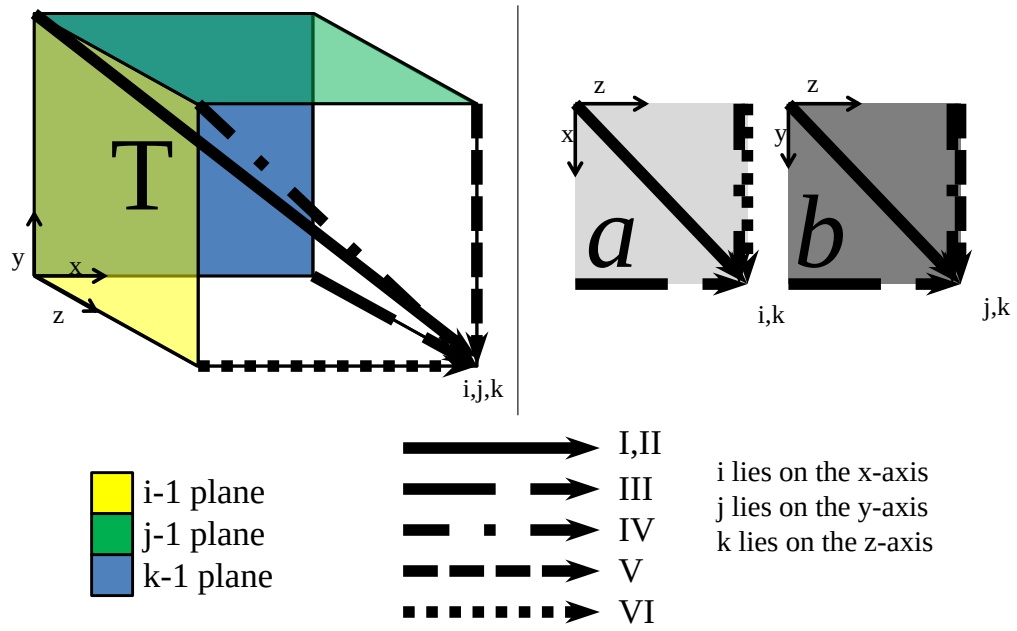

Figure 3.2.7.: The local dependencies of the calculation of a local similarity value. To the left in the three-dimensional table of accumulated similarity scores T and to the right the equivalent representation in the domain of the two sequences a and b. Operations V and VI – insertion and deletion – are one-dimensional operations and therefore appear only either in a or on b.

3.2.2 Contribution II: Locally Optimal Alignments

The state-of-the-art approach selects every position in the accumulated score table where the score is above a minimum similarity threshold as a candidate for a backtrack (BT) – and by extension as an alignment. This leads to a large number of alignments being calculated, especially if high-scoring alignments (i.e. large alignments with a high similarity density) are present in the data.

In such a high-scoring alignment, the minimal score is reached long before the end of the alignment. Each cell with a similarity score above this minimum generates a new BT and eventually the corresponding alignment, none of which are in any way meaningful. Alignments can branch in three (four, in the original algorithm) dimensions, which results in a very large number of BTs, as large *volumes* of cells containing high similarity values exist in the 3D scoring table. Whereas an individual BT operation is not extremely costly from a computational point of view, this high number of BTs effectively – and severely – limits either the usable problem size or the usable minimal accumulated similarity score (MASS).

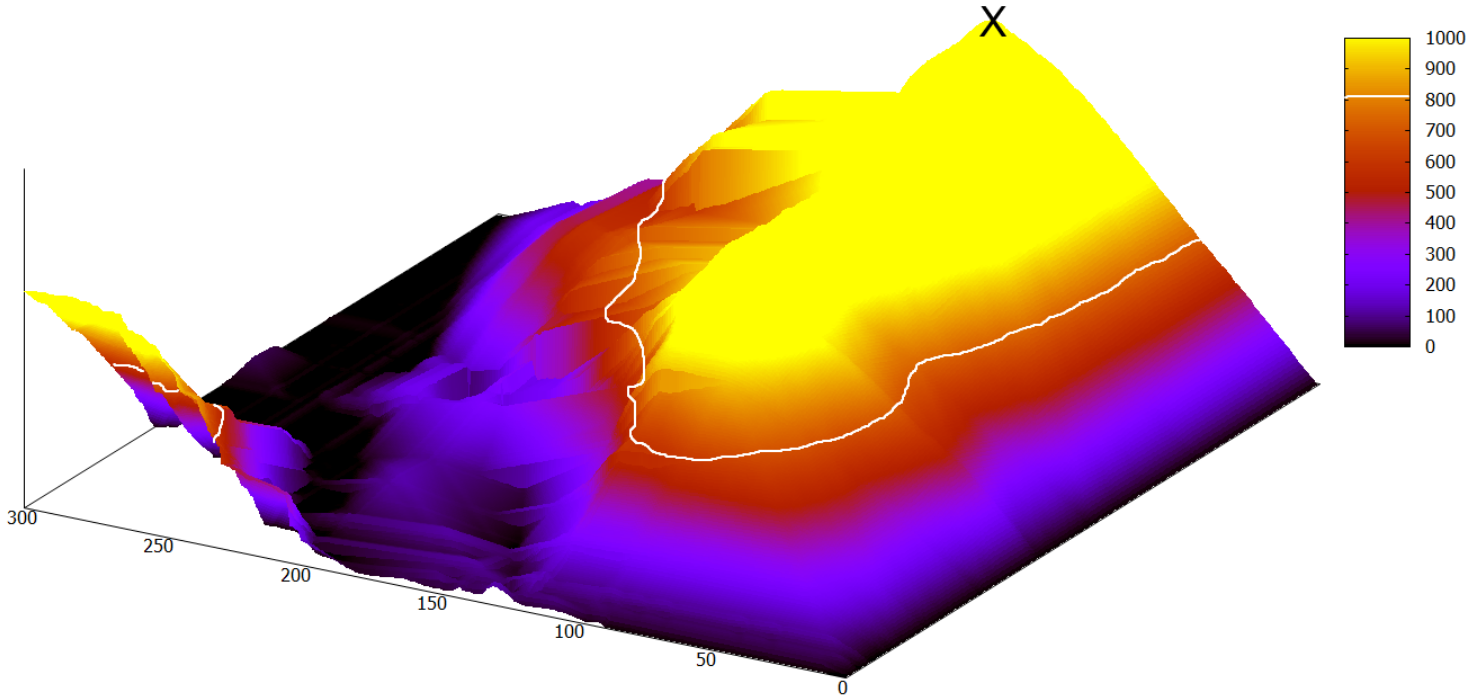

Figure 3.2.8.: The rightmost (n-th) z-slice of a table T. The axes of the plane correspond to the temporal axes of the sequences. The height and colour are representations of the accumulated similarity value in the cell in T. The white profile line denotes the plane of an arbitrarily chosen MASS. The highest value is denoted by an X.

In Fig. 3.2.8, we look at an extract of the n-th slice of the table T, where typically accumulated scores are highest compared to other z-slices. Any point above the reference plane at MASS-level is a potential candidate for a BT. This volume is visibly of considerable size.

To alleviate this problem, we retain only the most meaningful alignments. The inspiration for this choice can be found in bioinformatics, where the algorithm of Smith and Waterman [1981] is often used to identify *only* the optimum local alignment (Myers and Miller [1988]). This goes so far, that many evolutionary optimisations have focused exclusively on this problem. In fact, in Lecroq et al. [2012], reference is also made to a maximal element for the BT, but also in a global sense. Although this approach is too radical for our problem, we still perform a similar optimisation, but on a local scale.

We calculate BTs exclusively from the locally highest accumulated scores– as opposed to from every score higher than the MASS. This corresponds to the peak in Fig. 3.2.8.

A simple check whether any one of the 26 adjacent values is higher than the value in the current position determines if the current position in the accumulated score table is retained as a candidate for a BT. The number 26 stems from the fact that each cell has six directly neighbouring cells (one per surface of a cube), as well as three times four in-plane diagonal neighbours (one per vertex) and eight "node diagonal" neighbours (one per node), in the 3D table.

Although this introduces a large number of branches into the execution, it reduces the number of candidates (and therefore of expensive BTs) drastically. All of the removed backtracks are in essence redundant: The higher scoring alignment includes all elements a smaller alignment would include, and a larger alignment with a lower score would have an overall lower similarity, and therefore the added elements are not similar. We restrict ourselves to a maximum search radius of 1 ($\sqrt{2}$ for in-plane diagonals and $\sqrt{3}$ for space diagonals). Fig. 3.2.9 visualises how we obtain the number of 26 neighbours, and what these distances mean in the 3D geometry of the accumulated score table.

Though there is a benefit to extending the search radius, in that "double peaks" would no longer lead to twin alignments of largely similar nature, the added cost of extending the search radius would quadratically (surface of a sphere) increase the number of branches, and the additional reduction in candidates would be comparatively small. On the other hand, the number 26 is also the minimal number of checks required for this approach to be functional, as otherwise for large accumulated similarity values there would always be greater-than-MASS values in one of the adjacent positions.

For an illustration of the scale of this issue, Fig. 3.2.10 shows an example similarity score table. It shows that selecting a small MASS is necessary to detect smaller similar

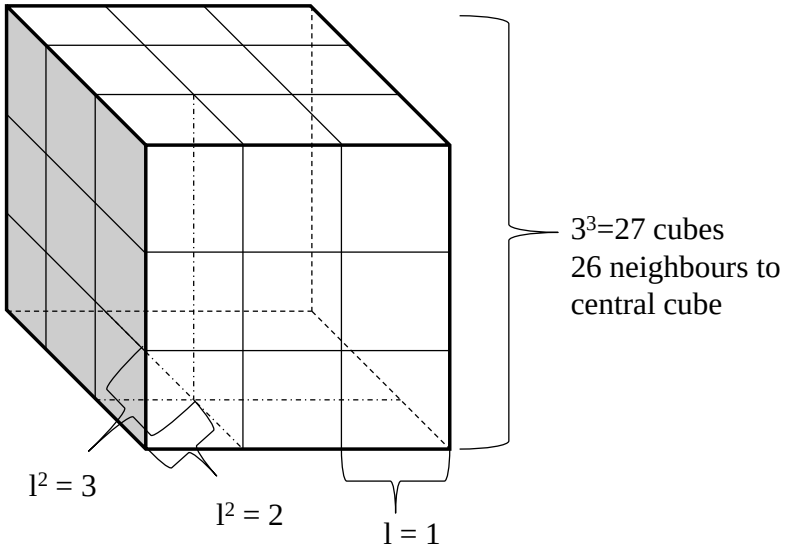

Figure 3.2.9.: The local neighbourhood of a node (in the centre of the shown cube) in the accumulated score table T. The shown cube consists of 27 sub cubes, the one at the centre being the "home node". The other 26 surrounding it, are the direct neighbours. A partial spatial subdivision is shown in the lower left corner to illustrate the notion of "radius".

structures (values above 1500 in this case are of significant interest), but large similar structures generate huge amounts of potential candidates at this smaller value. We reduce the volume of BTs from the volume around each of the peaks to just the number of peaks themselves. The side-by-side representation of the third dimension obscures somewhat that the 5 surfaces in the figure actually form a volume, but each point in a surface is neighbouring to the point in the same coordinates in a neighbouring surface, and they are all considered for the selection of final candidates for a BT.

3.3 Discussion

Reducing the degrees of freedom from the two-dimensional approach leads to a lowered complexity. Where originally the accumulated score table required O(lmn^2) (where l and m are the sequence lengths, n the number of elements in a tuple) space (and time to calculate), this has now been reduced to O(lmn), without any reduction in relevant capability. This means that we can now work with sequences that are n-times longer than previously, within the same system constraints.

The two-dimensional basis for the algorithm means that the order of columns still has

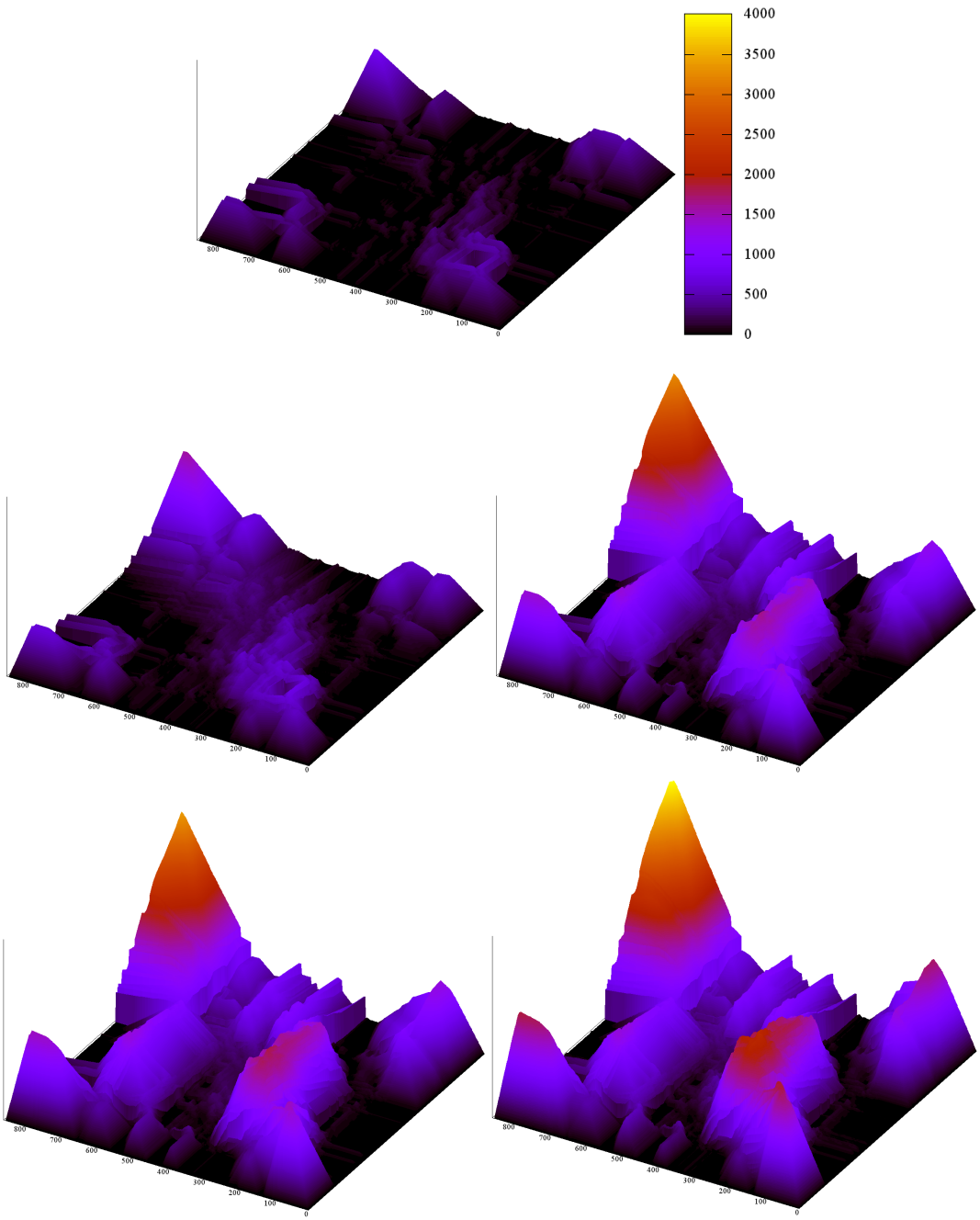

Figure 3.2.10.: Accumulated similarity score table for the alignment operations of the context data corresponding to two consecutive days. Each graph represents one of the five slices ($n = 5$) of the z-axis, starting at one at the top, incrementing to five at the bottom right. Height and colour correspond to local similarity values. The x - y plane is spanned by the temporal axes of the sequences. Note multiple peaks at different heights.

an impact on the accumulated similarity scores. A notion of single-step cursor movement is still present in our approach (cf. equation 3.2.1), and thus accumulated scores depend only and directly on the scores of the left tuple neighbour and the accumulated similarity score of the sequences to the left of the elements under consideration.

Reducing the number of BT candidates by local optimization can drastically increase performance, especially when the MASS is a small fraction of maximum accumulated similarity scores. If the neighbourhood of every peak consists of 5 cells in each direction of each of the three dimensions where the score is above MASS, this reduces the number of BTs by a factor of around 1000.

The average worst case (i.e. in an infinitely large table where cells with values higher than MASS are surrounded by exactly one layer of cells with value 0) improvement is by a factor of 7. This optimisation *does* remove some granularity, as possibly semantically atomic routine elements can be included within larger scale alignments, and thus disappear from view. On the other hand, they would be lost in the noise of meaningless alignments, if the classic approach were to be retained. A way to recover such smaller scale patterns, could be by iteratively locally aligning intervals of interest, with ever decreasing MASS.

Beyond this theoretical evaluation, we also performed an experimental validation. There we address whether the alignment approach is valid for context data, as opposed to just general sequential data.

Experimental Validation and Results

Overview

This part is dedicated to the evaluation of the alignment-based approach on both automatically annotated synthetic data and manually annotated real-world data. Before getting to the actual evaluation, we first examine the provenance of our test data. Consequently, we detail the synthetic data generation procedure. Additionally, we present the real world data collection campaign and the pre-treatment process required by our context data model. The evaluation of the algorithm on synthetic data is detailed in the following section, and the evaluation on real world data in the subsequent one.

4.1 Synthetic Data Evaluation

Testing on "real" data is crucial to being able to judge the "in the wild"-performance of an algorithm, but there are several limitations when solely relying on it. The manual annotation of ground truth is often labour intensive and error-prone. This limits the scope of possible evaluation. Conversely, an evaluation on synthetic data allows us to complement the results we can obtain from real world data, particularly by giving better control over the results and a wider variety of testing conditions.

4.1.1 Synthetic Data Generator and Dataset

In the following, we introduce a model and an algorithm that simulates parallel data from multiple sources containing cross-source repetitive patterns. Due to the many parameters that can be set, and the multiple random influences, the overall model for the generator is complex. For this reason, the description is split in five sub-sections,

of which the first lays out the requirements and design choices, the second some key definitions and terms used in the context of the description of the generation model and algorithm; the third and fourth sub-section detail each of the major sub-routines of generating data. An illustratory example is used throughout this subsection.

4.1.1.1 Data Generation Context Model

It is our goal to give the experimenter the greatest possible freedom with regard to the characteristics of the generated data. Some general expectations of what makes up context data guide us in the design of our generator. The basic structure of context is a sequence of n-tuples, when n context data sources are present. Within this sequence, we encounter two different states of context. Either the current context is part of a set of routine contexts, or it is not. A routine context influences a specific subset of context measurements, reproducibly each time it is present in the data. Per sensor, different amounts of data may be relevant to a pattern.

How much influence a routine context has on the context data is determined by how much variation there is between different instances of the context. This variation can be expressed as a random process. This random process modifies values between different instances of routine context, or leads to desynchronisation effects when elements are skipped or extended. It is inherent to each routine context. Furthermore, non-routine contexts may also lead to desynchronisation of following routine contexts, due to the global continuity of context. A separate random process determines these effects, and the amount of value modifications in the non-routine data, if a baseline has been defined. A model of probabilistic distances determines how likely it is for a certain value to be modified to another one, or to be extended or skipped.

The order of appearance of the routine contexts is usually regular as well, but can also vary under random influences. A real world example could be a person oversleeping, and skipping an entire morning activity, on their way to work. These varying orders can also be modelled by a random process.

Finally, different context sources may have interactions. As an example, location and radio signal strength are often correlated. Such correlations need to be taken into account when evaluating the random process which determines the ultimate values of a context. and may also be interesting when defining a set of routine contexts.

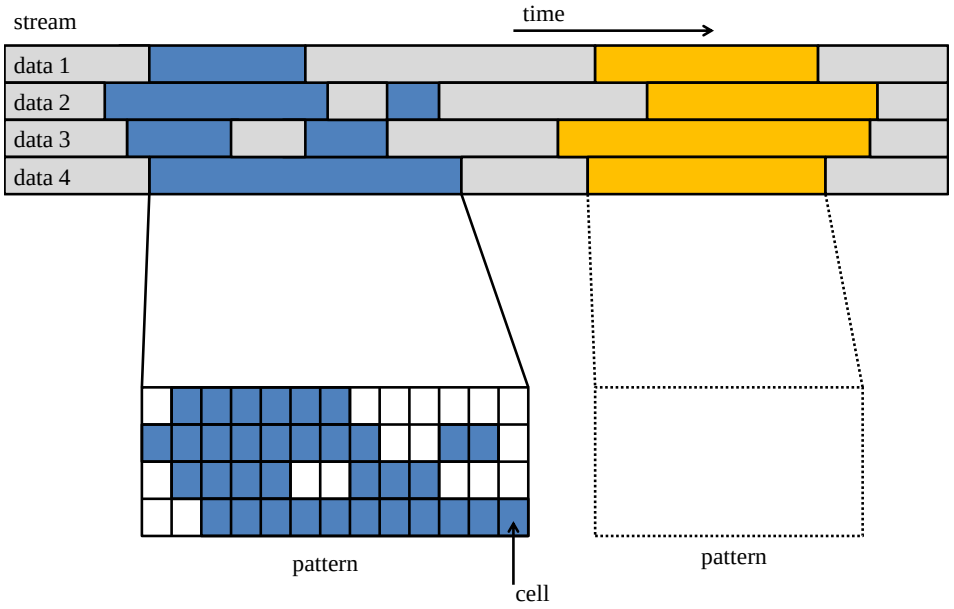

Figure 4.1.1.: Stream, pattern and cell for a simulated 4-tuple dataset. The stream consists of patterns (coloured) and random data (light grey). Each pattern consists of defined values (blue) and undefined values (white).

4.1.1.2 Definitions

The model used during the generation process consists of two key structures: the *stream* is a concept which represents data from multiple independent sensors evolving over time. In terms of the context model, it represents the sequence of n-tuples of sensor-data. *Patterns* are two-dimensional arrangements of data symbols in a rectangular grid (cf. the example in Fig. 4.1.1). The stream is created by alternating intervals of n-tuples of random data and randomly modified instances of these patterns. The output of the data is a direct representation of the stream.

The atomic unit of data – the individual grid element – is called a *cell*. Each cell is specific to a moment in time (horizontal) and a *data source* (vertical). Data sources in this context are discrete random variables, with a limited set of states ("*alphabet*"), each element of which is a *symbol*. The number of different attainable symbols - the cardinality of the alphabet - is called the *spectrum* of a data source (cf. Fig. 4.1.2). The real-world counter part to these data sources are filtered sensors producing discrete measurement values. Each cell of a pattern contains either a symbol from the corresponding alphabet or is a place holder asterisk "*****" value indicating an entry that is not defined by the pattern.

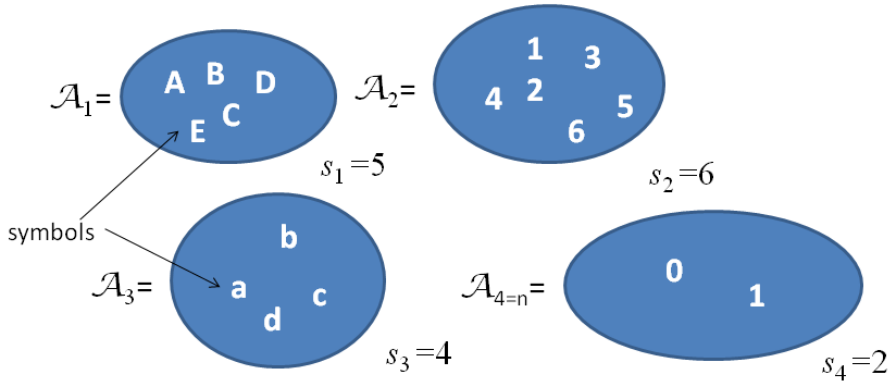

Figure 4.1.2.: Four alphabets $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3, \mathcal{A}_4$ with respective spectra 5, 6, 4 and 2. N.B.: Different symbols across different sensors (e.g. "A", "1", "a", "0") are used here for illustration purposes only, symbols are actually represented by their indices (i.e. as in $\mathcal{A}_2$).

Random effects – termed *noise* in the following – are a key factor in the generation of patterns and the stream. This noise is based on random variables, which are sampled, and the resulting values then transformed into length variations or symbol substitutions. All random variables that are used to generate noise are considered to be normally distributed (except in the limit-case of infinite variance, which is transformed into uniform distribution over an interval).

The effect of noise is derived from user defined *transition cost matrices* (one for each alphabet) with the number of rows and columns equal to the size of the alphabet, and *correlation matrices* (one for each pattern, and one for non-pattern intervals).

The transition cost matrices contain the cost of substitution between symbols. The costs of transitions for the following special symbols

***** place holder for empty cells in patterns;

del delete a cell from the stream;

ins add an additional value to the stream.

make up a further four vectors:

1. A vector containing the cost of conversion of any symbol *to* the ***** value;
2. A vector containing the cost of conversion *from* ***** to an alphabet value;

3. A vector associated to **ins**, which contains the cost of converting any alphabet value or * to an insertion;

4. A vector associated to **del**, which contains the cost of converting any alphabet value or * to a deletion.

These symbols form a global (i.e. across all data sources) alphabet of control-characters $\mathcal{A}_g = \{*, \mathbf{del}, \mathbf{ins}\}$. Of these, * can appear in patterns (hence there being both a "to" and "from" vector of transition costs), but none of these symbols appears in the stream. The operations linked to these symbols represent two different kind of noise effects: Temporal noise (**del** and **ins**) causes relative shifts in the time domain of data sources by inserting or deleting values, as well as local extension or compression of periods of data. Data noise (values replacing *) introduces random values in predefined areas, which corresponds to variable parts of otherwise fixed patterns and permits us to use rectangular patterns with little loss of generality.

The correlation matrices are lower triangular matrices with one line and column for each data source. Each line corresponds to the relative weights given to the calculation of a value by other values in the previous rows. A positive correlation value means that the influence of the random values used to determine the content of the respective cell have a quasi-linear effect on the determination of the current value. A negative correlation value conversely has an inverse quasi-linear effects. Zero values mark independent data sources. In this context "quasi-linearity" is an artificial effect to transform multi-dimensional random values from one space to another. This may not hold up to a comparison to real world data, but provides a reasonably simple model which avoids having to define correlations per pattern and per symbol individually.

The following naming conventions are used henceforth: $\mathbb{Z}_{>0} = \{1, 2, 3, \ldots\}$ the natural numbers excluding zero and $\mathbb{R}_{\geq 0} = \bigcup_{x \in \mathbb{R}, x \geq 0} \{x\}$ all non-negative real values. The notation $x_{i,h,k}$ represents the element in the h-th line of the k-th column of a matrix (or table) X_i.

Let $n \in \mathbb{Z}_{>0}$ be the number of data sources, $m \in \mathbb{Z}_{>0}$ be the number of patterns, $\mathcal{A}_i = \{x \in \mathbb{Z}_{>0} | x \leq s_i\}, (1 \leq i \leq n)$ be the (abstracted) alphabet of the i-th data source and $l_j \in \mathbb{Z}_{>0}$ be the length of the j-th pattern $X_j \in \mathfrak{P} \subset \bigcup_{j=1}^{m} \bigtimes_{i=1}^{n} (\mathcal{A}_i \cup \{*\})^{l_j}$ of the indexed set of patterns (cf. Fig 4.1.3 for a complete set of patterns, with empty cells, a sample elements and corresponding dimensions.). Let $\vec{s} \in \mathbb{N}_{>0}^n$ with components s_i be the spectra-vector where $\forall_{1 \leq i \leq n} : s_i = |\mathcal{A}_i|$, with $|\cdot|$ being the cardinality and let $l_{\text{stream}} \in \mathbb{Z}_{>0}$ be the number of total grid columns of the stream.

In the following, $x, y \in \mathcal{A}_i \cup \mathcal{A}_g, (1 \leq i \leq n)$ stand for symbols.

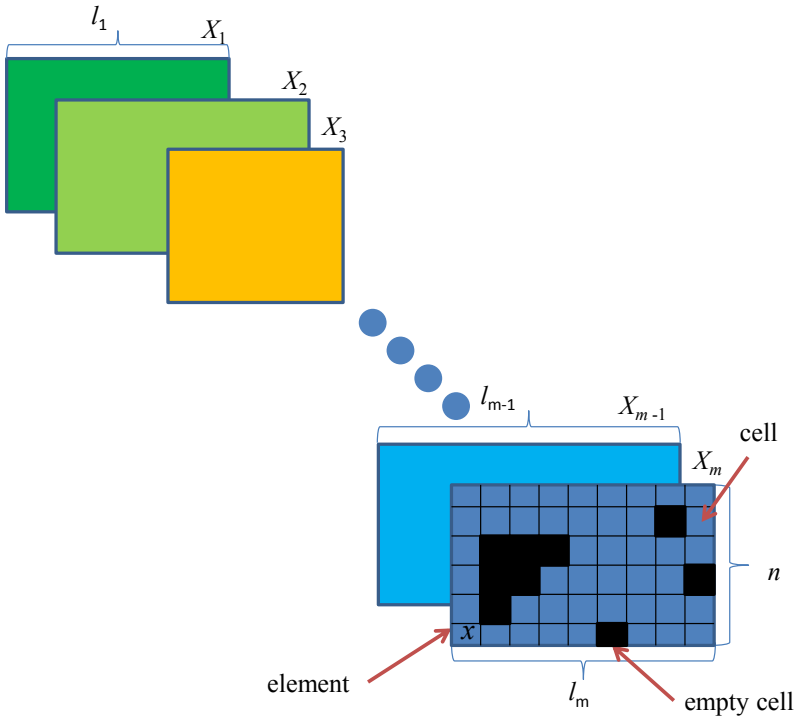

Figure 4.1.3.: A set of patterns, $\mathfrak{P} = \{X_1, \ldots, X_m\}$, with n data sources and lengths $l_1, \ldots, l_m$, and a pattern element $x = X_{m,n,1} \in \mathcal{A}_n$.

$\mathcal{N}(\sigma_\eta), \eta \in \{o, p, q, r, u\}$ are Gaussian distributions with variance σ_η^2 serving as sources of random values for

o value variety of noise intervals between two instances of patterns in the stream, random variable $\mathfrak{o}$;

p noise applied to pattern-defined values during the instantiation of patterns in the stream, random variable $\mathfrak{p}$;

q variations in length – unless lengths are predefined for each pattern – and values between two different patterns, random variable $\mathfrak{q}$;

r variations of the interval length between two instances of the same pattern in the stream, random variable $\mathfrak{r}$;

u variations of the length of noise intervals in the stream, random variable $\mathfrak{u}$.

The transition cost matrices are $T_i = [t_{i,h,k}], (1 \leq i \leq n) \in \mathbb{R}_{\geq 0}^{s_i \times s_i}$ and form the set $\mathcal{T} := \bigcup_{i=1,}^{n} \{T_i\}$. The entries $t_{i,h,k}$ are the positive real-valued costs of transforming the

h-th symbol of $\mathcal{A}_i$ into the k-th symbol. The costs of transition relative to the elements of $\mathcal{A}_g$ are the pattern insertion and deletion cost vectors $\vec{b}_i$, $\vec{c}_i \in \mathbb{R}^{s_i}$ for the transitions from and to *****, and the stream insertion and deletion cost vectors $\vec{d}_i, \vec{e}_i \in \mathbb{R}^{(s_i+1)}$ for transitions to **del** and **ins**.

Transition cost tables are related to the substitution score tables of the algorithms used to align pattern instances. In the alignment algorithm, they indicate similarity based on the way the model is configured. In this generator, they *define* the probability of random transformations of symbols. An important feature of transition cost tables is that they do not necessarily need to be symmetric, even though a naive approach considering the value spaces as Euclidean spaces with distances would indicate this. Yet, when using probabilities of transition in example data as basis for the cost of state-transitions, the assumption of symmetry is rendered invalid, as the resulting graph of transitions is not necessarily symmetric. For the values in the table to have the expected effect, they need to be scaled in consideration of the variances of $\mathfrak{p}$ and $\mathfrak{q}$.

A gappiness vector $\vec{g} \in \mathbb{R}^n_{\geq 0}$, with $\forall^n_{i=1} \vec{g}_i \leq 1$ contains the ratio of pattern entries to empty cells for each data source, and simulates effects of limited data source availability and information significance and density.

Let lower triangular matrices $\xi_i \in \mathbb{R}^{n \times n}, i = \{1, .., m\}$ be the correlation matrices for patterns X_i and let ξ_0 be the correlation matrix for non-patterned values. These matrices are lower triangular, as each new value can only ever be correlated to values that have already been generated. The entries are normalized so that each row-sum is equal to one.

Let $\vec{w} \in \mathbb{Z}^m_{>0}$ be a representation of the frequency of pattern apparition in the stream. Each value corresponds roughly to the relative period of apparition of a pattern in the stream. Finally, let $l_{\text{noise}} \in \mathbb{R}$ be the average length and let $\bar{l}_{\text{noise}} \in \mathbb{Z}_{>0}$ be the maximum length of a non-pattern-interval in the stream.

For our running example, let $n = 4$, $m = 3$, $\vec{s}^\top = (5, 6, 4, 2)$ and $\vec{l}^\top = (10, 6, 7)$. Let the corresponding gappiness vector be $\vec{g} = (0.1, 0.1, 0.3, 0.3)$, and let the first of four transition tables be

$$T_1 = \begin{pmatrix} 0 & 7 & 5 & 4 & 8 \\ 7 & 0 & 2 & 5 & 3 \\ 5 & 2 & 0 & 1 & 2 \\ 4 & 5 & 1 & 0 & 3 \\ 8 & 3 & 2 & 3 & 0 \end{pmatrix} \in \mathcal{T}.$$

Furthermore, let

$$\vec{b}_1 = \begin{pmatrix} 3 \\ 9 \\ 5 \\ 9 \\ 2 \end{pmatrix}, \vec{c}_1 = \begin{pmatrix} 4 \\ 8 \\ 3 \\ 4 \\ 9 \end{pmatrix}, \vec{d}_1 = \begin{pmatrix} 5 \\ 2 \\ 3 \\ 9 \\ 7 \\ 4 \end{pmatrix}, \vec{e}_1 = \begin{pmatrix} 5 \\ 2 \\ 4 \\ 7 \\ 6 \\ 6 \end{pmatrix}$$

and the correlation matrices

$$\xi_0 = \xi_1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0.5 \end{pmatrix}.$$

Finally, let $\vec{w}^\top = (45{,}30,\ 50)$, $l_{\text{noise}} = 4.8$ and $\bar{l}_{\text{noise}} = 15$.

We use these values to demonstrate how to generate pattern values and pattern instances in the stream.

4.1.1.3 Pattern Generation

The first step R_1 of the algorithm is to generate m different patterns according to the parameters given:

$$X_1 = R_1(n, l_1, \vec{s}, \mathcal{T}, \xi_1, \vec{g})$$

$$X_j = R_1(n, l_j, \vec{s}, X_1, \sigma_q, \mathcal{T}, \xi_j),\ j = 2, \ldots, m$$

A reference pattern X_1 is generated first, consisting of randomly selected entries based on vectors $\vec{\gamma}_l \in \bigtimes_{i<n}]0, s_i], l = 1, \ldots, l_1$ of uniformly distributed continuous random values. The other patterns are derived from this pattern and the inter-pattern similarity parameters $0 < \sigma_q < \infty$.

Before a cell is filled, a random test (uniformly distributed random variable over the interval $[0, 1]$) against $\vec{g}$ determines whether a cell contains an alphabet symbol. If the random value is less than the gappiness quotient, the cell is filled with a *-value,

otherwise the following formula is used to obtain the value:

$$x_{1,k,l} = \lceil \sum_{j \leq k} \frac{\xi_{1,j,k} s_k \gamma_{l,j}}{s_j} \rceil - \lfloor \frac{\lceil \sum_{j \leq k} \frac{\xi_{1,j,k} s_k \gamma_{l,j}}{s_j} \rceil}{s_k} \rfloor s_k$$

$$\Longleftrightarrow$$

$$x_{1,k,l} \equiv \lceil \sum_{j \leq k} \frac{\xi_{1,j,k} s_k \gamma_{l,j}}{s_j} \rceil (\text{mod } s_k),\ l = 1, \ldots, l_1$$

This calculates the value in a cell as the ceiling of the sum of the correlation-weighted random values, and uses a modulo with the spectrum to "roll-over" in case of overflows.

For our running example, this means the following: For sensor one, a uniformly distributed random variable over the continuum $[0, 1]$ is evaluated. If this value is larger than $g_1 = 0.1$ then $x_{1,1,1} = \lceil \gamma_{1,1} \rceil$, otherwise $x_{1,1,1} = *$. In the following, we shall assume that $\gamma_{1,1} = 1.935$ and thus $x_{1,1,1} = 2$.

To calculate $x_{1,4,1}$, the first entry of the fourth sensor, $\gamma_{1,1}$ is requireds, because the corresponding row in ξ_1 is $(0.5, 0, 0, 0.5)$ with a non-zero first component. This means that the value is calculated using the previously introduced $\gamma_{1,1} = 1.935$ and a newly determined $\gamma_{1,4}$ which we assume to be 0.478 for this calculation. Then,

$$\begin{aligned}
x_{1,4,1} &= \lceil \sum_{j \leq 4} \frac{\xi_{1,j,4} s_4 \vec{\gamma_{1,j}}}{s_j} \rceil - \lfloor \frac{\lceil \sum_{j \leq 4} \frac{\xi_{1,j,4} s_4 \vec{\gamma_{1,j}}}{s_j} \rceil}{s_4} \rfloor s_4 \\
&= \lceil \sum_{j \leq 4} \frac{\xi_{1,j,4} 2 \vec{\gamma_{1,j}}}{s_j} \rceil - \lfloor \frac{\lceil \sum_{j \leq 4} \frac{\xi_{1,j,4} 2 \vec{\gamma_{1,j}}}{s_j} \rceil}{2} \rfloor 2 \\
&= \underbrace{\lceil \frac{0.5 \cdot 2 \gamma_{1,1}}{5} + \ldots + \frac{0.5 \cdot 2 \gamma_{1,4}}{2} \rceil}_{(\circ)} - \lfloor \frac{\circ}{2} \rfloor 2 \\
&= \lceil \frac{0.5 \cdot 2 \cdot 1.935}{5} + \frac{0.5 \cdot 2 \cdot 0.478}{2} \rceil - \lfloor \frac{\circ}{2} \rfloor 2 \\
&= \lceil 0.387 + 0.239 \rceil - \lfloor \frac{\circ}{2} \rfloor 2 = 1 - \lfloor \frac{1}{2} \rfloor 2 = 1
\end{aligned}$$

The other patterns are determined in two ways, depending on σ_q:
If $\sigma_q = \infty$, patterns $X_2, \ldots, X_m$ are determined in the same way as the reference pattern and can be considered to be pair-wise independent of one another, with the exception of possible interactions due to correlation matrices. $\sigma_q \gg \max(\mathcal{T})$ where $\max(\mathcal{T})$ denotes the maximum transition cost, has a very similar effect.
In the case of a finite $0 < \sigma_q \in \mathbb{R}$ (and ideally smaller than the maximum transition-cost) patterns $X_{j,2 \leq j \leq m} \sim X_1$ (where $\sim$ indicates similarity between two matrices) are

constructed by creating a set of n vectors $\vec{\psi_i} \in \mathbb{R}^{s_i+1}$ of random real values which are distributed according to $\mathcal{N}(\sigma_q)$. The vectors $\vec{\psi_i}, i > 1$ are then modified to reflect the correlations dictated by ξ_1:

$$\psi'_{i,l} := \sum_{k=1}^{i} \xi_{1,i,k} \frac{s_i + 1}{s_k + 1} \sum_{j=\lfloor \frac{(l-1)(s_k+1)}{s_i+1} \rfloor + 1}^{\lceil \frac{l(s_k+1)}{s_i+1} \rceil} \vec{\psi}_{k,j} \omega_{i,j,k} \tag{4.1.1}$$

where

$$\omega_{i,j,k} := \begin{cases} j - \frac{(l-1)(s_k+1)}{s_i+1} & \text{if } j < \frac{(l-1)(s_k+1)}{s_i+1} + 1 \wedge s_k > s_i \\ \frac{l(s_k+1)}{s_i+1} - j + 1 & \text{if } j > \frac{l(s_k+1)}{s_i+1} \wedge s_k > s_i \\ 1 & \text{if } \frac{(l-1)(s_k+1)}{s_i+1} + 1 \leq j \leq \frac{l(s_k+1)}{s_i+1} \\ \frac{l(s_k+1)}{s_i+1} - j + 1 & \text{if } j > \frac{(l-1)(s_k+1)}{s_i+1} + 1 \wedge s_k < s_i \\ j - \frac{(l-1)(s_k+1)}{s_i+1} & \text{if } j < \frac{l(s_k+1)}{s_i+1} \wedge s_k < s_i \\ \frac{s_k+1}{s_i+1} & \text{if } \frac{l(s_k+1)}{s_i+1} \leq j \leq \frac{(l-1)(s_k+1)}{s_i+1} + 1 \end{cases}$$

which corresponds to a constant resampling, linearising across vector dimensions by treating them as intervals. An example is given in Fig. 4.1.4. The values on each arrow correspond to the scaled value that is used to determine the value at the end of the arrow. In the left case, the values in the left vector are scaled up (by 5/3) and in the right case, the values are scaled down (by 3/5). Then, these scaled values are spread according to the coverage they have on the corresponding parts of the vector. In the case of the middle value (2 - marked with red dashed line) in the left example, this is 1/5 for the second value of the right vector, 1 for the third value, and 1/5 for the fourth value. Scaling is applied accordingly, hence $(2 \cdot 5/3 \cdot 1/5) = 2/3$ is the influence on the top value of the three values that are marked in the right vector.

Let $x = x_{1,i,l}$ be the value of the i-th row and l-th column of X_1 and the k-th element of $\mathcal{A}_i$. Let $v_x \in \mathbb{R}^{s_i+1}$ be a vector whose components are the l-th of T_i and the l-th element of $\vec{c}$:

$$v_x := \begin{pmatrix} t_{i,1,k} \\ t_{i,2,k} \\ \vdots \\ t_{i,s_i,k} \\ c_{i,k} \end{pmatrix}$$

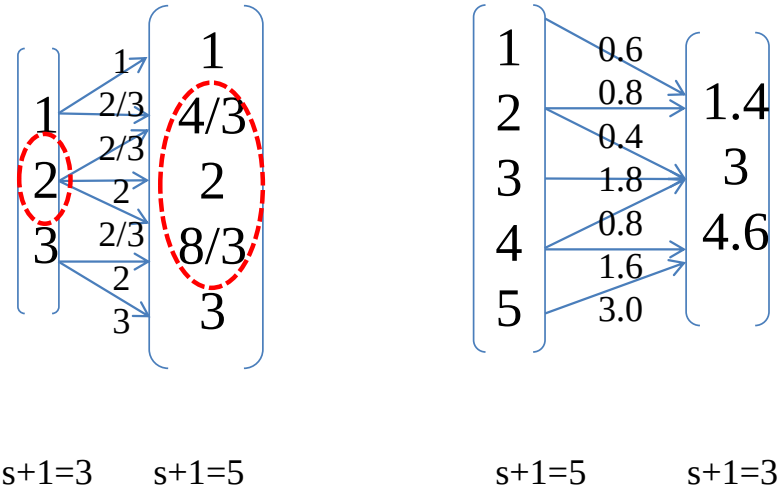

Figure 4.1.4.: Illustration of dependence coefficient determination during creation of similar patterns. The right vectors are created from the values of the left vectors, through linear interpolation and scaling.

or, in the case of $x = *$:

$$v_* := \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \\ 0 \end{pmatrix}$$

The index y of the minimal component of $|v_x - \vec{\psi}_i'|$,

$$y = \min_\sharp |v_x - \vec{\psi}_i'|$$

(here $|\cdot|$ is the component-wise absolute value) is the value that takes the place of x in X_j, unless $y = s_{i+1}$ in which case * is inserted into the pattern at this position. This is repeated for all $n \times l_j$ entries of the pattern.

Furthermore, the length l_j of the j-th pattern – if not specifically set to a certain value beforehand – is calculated by obtaining a random real value Δl_j from $\mathcal{N}(\sigma_q)$, adding it to l_1 and rounding to the closest integer:

$$l_j := \begin{cases} \lfloor \Delta l_j + l_1 + 0.5 \rfloor & \Delta l_j + l_1 > 0 \\ 0 & \Delta l_j + l_1 = 0 \\ \lceil \Delta l_j + l_1 - 0.5 \rceil & \Delta l_j + l_1 < 0 \end{cases}$$

Addition or removal of elements is done column-wise: for every column, a random check is performed against $\frac{|\Delta l_j|}{\max(l_j, l_1)}$ (until the l_j-th column is reached) to determine whether the current column of X_1 is skipped or a column of equally distributed randomly selected symbols inserted.

To compute $x_{2,1,1}$ of the pattern $X_2 \in \mathbb{Z}_{>0}^{4\times 6}$ of our running example, given $\sigma_q^2 = 2$ as the variance of the distribution of $\mathfrak{q}$, we first determine whether the first line is skipped to make up for the difference in length to X_1. This is done by obtaining a random value from the interval $[0, 1]$ and testing whether it is smaller than $\frac{10-7}{\max(10,7)} = 0.3$. We assume – for the sake of this example – that this is not the case, and instead $x_{2,1,1}$ is derived from $x_{1,1,1}$. We obtain a vector $\vec{\psi_1} \in \mathbb{R}^6$ by repeatedly sampling $\mathfrak{q}$: $\psi_1^\top = (0.1, 0.5, -1.6, 1.1, -0.8, 0.3)$. Then

$$\begin{aligned} x_{2,1,1} &= \min_\sharp |v_{x_{1,1,1}} - \vec{\psi_1'}| = \min_\sharp |v_2 - \vec{\psi_1}| \\ &= \min_\sharp \begin{pmatrix} t_{1,1,2} \\ t_{1,2,2} \\ t_{1,3,2} \\ t_{1,4,2} \\ t_{1,5,2} \\ c_{1,2} \end{pmatrix} - \vec{\psi_1} = \min_\sharp \begin{pmatrix} 7 - 0.1 \\ \mathbf{0 - 0.5} \\ 2 + 1.6 \\ 5 - 1.6 \\ 3 + 0.8 \\ 8 - 0.3 \end{pmatrix} = 2 \end{aligned}$$

The values $x_{2,k,l}, (k > 1)$, are calculated by taking into account the correlation matrix. Let $\vec{\psi_4}^\top = (-1.5, 0.8, -0.3)$. We determine $\vec{\psi_4'}$ using equation 4.1.1, and the resampling of $\vec{\psi_1}$ shown in Figure 4.1.4:

$$\vec{\psi_4'} = 0.5 \begin{pmatrix} 0.3 \\ -0.25 \\ -0.5 \end{pmatrix} + \ldots + 0.5 \begin{pmatrix} -1.5 \\ 0.8 \\ -0.3 \end{pmatrix} = \begin{pmatrix} -0.6 \\ 0.275 \\ -0.4 \end{pmatrix}$$

This allows us to calculate

$$x_{2,4,1} = |\min_\sharp \vec{\psi_4'} - v_{x_{1,4,1}}|$$

as above.

These steps are repeated until m patterns have been created. These patterns are then integrated into the stream.

4.1.1.4 Stream Generation

Streams are generated by interposing modified instances of patterns with blocks of random values. The generation of streams can be considered as a relation

$$R_2(n, \mathcal{T}, \mathfrak{P}, \sigma_p, \sigma_o, \sigma_r, \sigma_u, \vec{w}, l_{\text{noise}}, \bar{l}_{\text{noise}}, \xi_0) :$$
$$\mathbb{Z}_{>0} \times \mathbb{R}^{n \times s_i \times s_i} \times \bigcup_{j=1}^{m} \bigtimes_{i=1}^{n} (\mathcal{A}_i^{l_j} \cup \{*\}) \times \mathbb{R}_{\geq 0}^{n} \times \mathbb{R}_{\geq 0}^{n} \times \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \times \mathbb{Z}_{>0}^{m} \times \mathbb{R} \times \mathbb{Z}_{>0}$$
$$\to \bigtimes_{i=1}^{n} (\mathbb{Z}_{s_i+1} \setminus \{0\})^{l_{\text{stream}}}$$

This is a projection of the configuration space (dimensions, transition model, patterns, randomness, correlations) into a stream. There are three main tasks to consider:

Scheduling Each pattern has a predefined frequency of instantiation, which can be understood as an m-vector $\vec{w} \in \mathbb{R}^m$. σ_p determines how much the appearance intervals vary.

Noise Noise duration and variety are defined by $l_{noise}, \bar{l}_{noise}, \sigma_u$ and σ_o.

Modifications The variance between instances of the same pattern is defined by $\sigma_r \in \mathbb{R}^n$.

Scheduling is managed in the following way:

1. During the first run, $\vec{w}' = \vec{w}$ is created in order to track changes to $\vec{w}$ without losing the original information.

2. The pattern $X_{\min_\sharp(\vec{w}')}$ is selected and inserted into the stream, where $\min_\sharp(\vec{w}')$ is the index of the minimal component of $\vec{w}'$:

$$\min_\sharp(\vec{w}') = i | \forall_{j=1}^{|\vec{w}'|} : w_i' \leq w_j'.$$

   If multiple components share the same smallest value, the value of $\min_\sharp$ is picked at random from the corresponding indices.

3. The vector $\vec{w}'$ is updated by adding $\vec{w}_{\min_\sharp(\vec{w}')} + \mathfrak{r}$ to the component pertaining to $X_{\min_\sharp(\vec{w}')}$: $w'_{\min_\sharp(\vec{w}')} = w'_{\min_\sharp(\vec{w}')} + w_{\min_\sharp(\vec{w}')} + \mathfrak{r}$

Algorithm 4.1 Scheduling pattern instances

input : The scheduling vector $\vec{w} \in \mathbb{R}^m$
output: A scheduled stream

$\vec{w}' \leftarrow \vec{w}$;
$l \leftarrow 0$;
while $l < l_{stream}$ **do**
  $j_{\text{ins}} \leftarrow \min_\sharp(\vec{w}') = i | \forall_{j=1}^{|\vec{w}'|} : w'_i \leq w'_j$;
  append pattern $X_{j_{\text{ins}}}$ to stream;
  increment l by the length of $X_{j_{\text{ins}}}$;
  $w'_{j_{\text{ins}}} \leftarrow w'_{j_{\text{ins}}} + w_{j_{\text{ins}}} + \mathfrak{r}$; `// update minimal value in` $\vec{w}'$
  for $j \leftarrow 1$ **to** m **do** `// update other values in` $\vec{w}'$
    if $j \neq j_{ins}$ **then**
      $w'_j \leftarrow w'_j - \frac{w_{j_{\text{ins}}}}{m-1}$
    end
  end
  append noise interval of length $\min((l_{\text{noise}} + \mathfrak{u}), \bar{l}_{\text{noise}})$ to stream;
  $l \leftarrow l + \min((l_{\text{noise}} + \mathfrak{u}), \bar{l}_{\text{noise}})$
end

4. $\forall_{j=1,\ldots,m, j \neq \min_\sharp(\vec{w}')} : w'_j = w'_j - \frac{w_{\min_\sharp(\vec{w}')}}{m-1}$, which avoids under- and overflows due to incrementation and decrementation, as the decrement of each step is equal to its increment, and $E(\mathfrak{r}) = 0$.

5. A noise interval of the length of $\min((l_{\text{noise}} + \mathfrak{u}), \bar{l}_{\text{noise}})$ is injected into the stream, containing random values that adhere to the dependencies defined by ξ_0.

This is also formulated in pseudocode in algorithm 4.1.

Within the frame of our example, this has the following effects: First, an interval of noise of length 5 is appended to the stream based on the assumption that the average value of 4.8 is not significantly impacted by the random effects of $\mathfrak{u}$. Then, pattern $X_{\mathbf{2}}$ is appended to the stream, as the smallest value in $\vec{w}^\top$ (30), is in the second position. $\vec{w}'$is then updated as follows:

$$\vec{w}' = \begin{pmatrix} w'_1 - \frac{w_2}{2} \\ w'_2 + w_2 + \mathfrak{r} \\ w'_3 - \frac{w_2}{3} \end{pmatrix} = \begin{pmatrix} 45 - 15 \\ 30 + 30 + \mathfrak{r} \\ 50 - 15 \end{pmatrix} = \begin{pmatrix} 30 \\ 60 + \mathfrak{r} \\ 35 \end{pmatrix}$$

The contents of a noise interval cell are generated by determining the index of the smallest component of the distance $\Delta_{\mathfrak{o},i} = |\phi_i - \hat{v}_0|$, where $\phi_i \in \mathbb{R}^{s_i+2}$ is a set of vectors of subsequent results of the event at the base of $\mathfrak{o}$, modified according to the method

proposed in equation 4.1.1 (replacing ξ_1 with ξ_0 and adjusting sizes) and $\hat{v}_0 \in \mathbb{R}^{s_i+2}$ is $\vec{b}$ appended by the last entries of $\vec{d}$ and $\vec{e}$ respectively:

$$\hat{v}_0 = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{s_i} \\ d_{s_i+1} \\ e_{s_i+1} \end{pmatrix}$$

Thus, the symbol added to the stream is:

$$y = \min_\sharp |\phi - \hat{v}_0|$$

In the case of $y = s_i + 2$, the control value **ins** is generated: a new ϕ is randomly obtained, $\Delta_{\mathfrak{o},i}$ re-evaluated, and a new value is inserted after the current position using this very same algorithm. If the minimum index obtained is $s_i + 1$, a **del** control value is generated and no value is written into the i-th row of the stream during this iteration. For all other indices, the resulting index corresponds directly to the symbol of the corresponding alphabet to be written into the stream. Once this is done, the algorithm continues, by performing the same actions on the symbol in the cell to the right, for all columns that are to be generated.

The modifications applied to instances of patterns in the stream are calculated in the same way for non-defined cells. For cells of patterns containing symbols, the above algorithm is adapted by calculating $\min_\sharp \Delta_{\mathfrak{p},i} = \min_\sharp |\rho - \hat{v}_x|$, with $\rho \in \mathbb{R}^{s_i+2}$ a vector of random values obtained by sampling $\mathfrak{p}$ $(s_i + 2)$-times, and x being the value in the pattern cell and hence $\hat{v}_x$ being the x-th column of T_i appended by the x-th elements of $\vec{d}$ and $\vec{e}$:

$$\hat{v}_x := \begin{pmatrix} t_{i,1,x} \\ t_{i,2,x} \\ \vdots \\ t_{i,s_i,x} \\ d_x \\ e_x \end{pmatrix}$$

In Figure 4.1.5, we show how the element $x_{2,1,1} = 2$ is instantiated into the stream, modified by noise to become the sensor value 4.

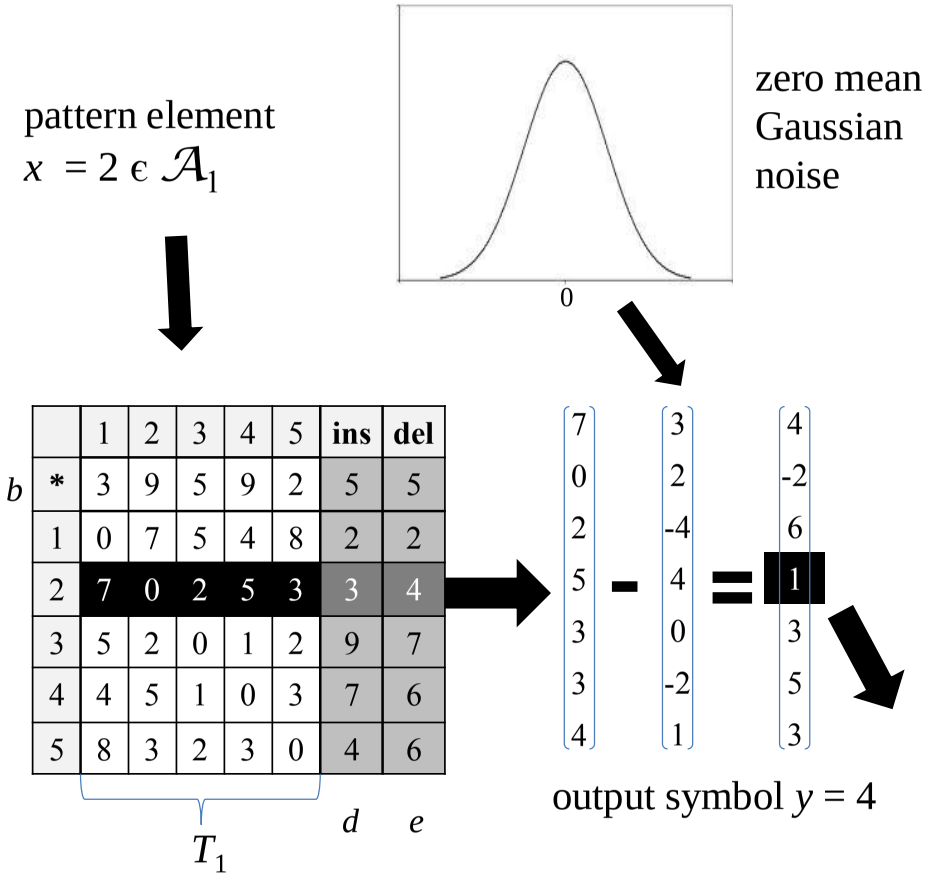

Figure 4.1.5.: Noise is applied to symbol x from alphabet $\mathcal{A}_1$ using transition cost matrix T_1 and transition cost vectors $\vec{b_1}, \vec{d_1}, \vec{e_1}$. The resulting value is $y = 4$. This example uses integer values for easier reading and comprehension, in general the random values are real values. The process starts with value $x_{2,1,1} = 2$ from pattern X_2, extracts the corresponding row from T_1 and selects the smallest absolute value in the sum of the extracted row and a random vector, to determine the index of the element to replace x.

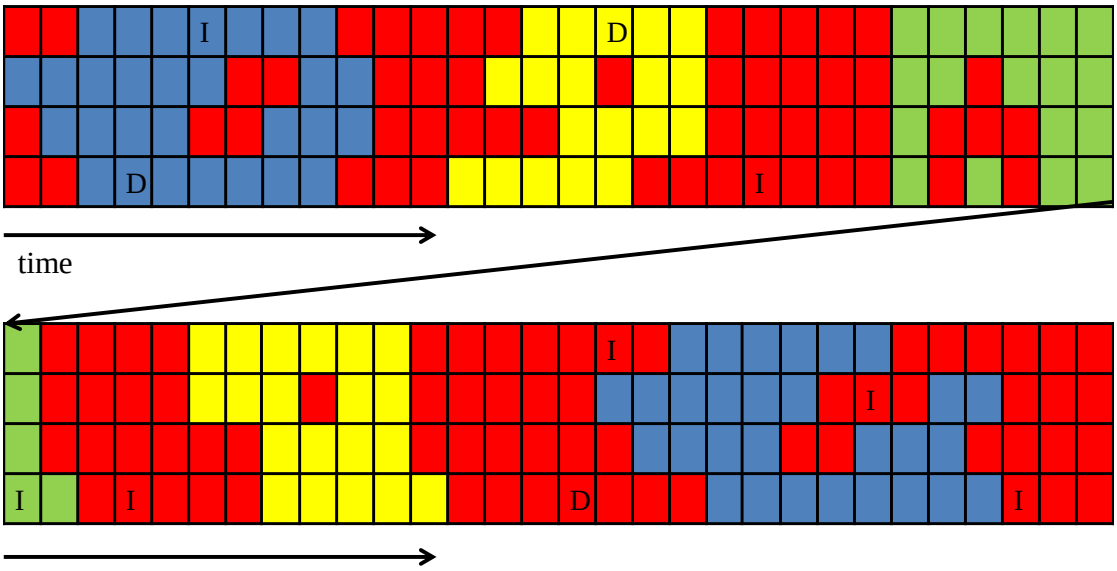

Figure 4.1.6.: Extract of a possible resulting stream with $n = 4$ and $m = 3$ different patterns (blue, yellow and green) and noise (red). I and D indicate cells in which insertions or deletions have happened. y is the symbol that has been generated in Fig. 4.1.5.

This process (schedule - noise - pattern) is repeated until the sum over all noise-interval-lengths and pattern-instance-lengths is equal or larger than l_{stream}. A schematic example of a resulting stream is given in Fig. 4.1.6.

With regard to our initially targeted model, this allows us to control most variables precisely. We can generate data that locally or progressively desynchronises, data that has controlled amounts of noisy variation and in any size or shape desirable. To help this latter fact, we also permit the handcrafting and loading of pre-defined patterns.

We currently identify the following weaknesses to our approach: we limit ourselves to normally distributed noise for all random aspects of the generation process. We consider this as a safe default choice, especially to model sensor measurement noise, but it may not be an accurate model for variations caused by human actions. In the absence of a better model for this kind of variation, we restricted ourself to Gaussian distributions.

Our correlation algorithm between two data sources with different numbers of symbols is not correct, in the sense that we linearise across dimensions which have no actual linear relationship. On the other hand, we see this as the only way to implement correlation. Due to the difficult nature of this feature, we do not use it for the generation of our data, based on the assumption that strongly correlated data is unified to a single sensor reading in real world context data.

A final problematic issue is that of our multidimensional noise issue, which makes the link between a chosen variance value and the actual effect on value transitions rather unintuitive. Although a probability interval based approach may have had more predictable results, it would be more complex to integrate with the notion of data source correlation, and would require a rather complex calculation of interval limits for each of the possible transitions.

Taking into account these limitations, we feel nonetheless confident that it allows us to generate a number of well understood datasets. The ability to retain the information of which pattern is instantiated in which cells of the output stream allows us to evaluate our alignment algorithm against this ground truth.

4.1.1.5 Dataset Generation

For the evaluation procedure, we generate 135 datasets, which can be characterised by five different scenarios, each of which has 27 different variations by adjusting three variables. Each scenario serves to link a configuration of the data generator, to a specific type of behaviour of a simulated human exhibiting a certain way of life. These five scenarios are:

1. A scenario without random influences, outside the order of pattern instantiation. This corresponds to a human who reproduces the exact same set of context data every time a certain activity is performed, and always performs activities that are repeated eventually.

2. A scenario with an interval (of length 10) of random data between each two pattern instances. This could represent a person that performs some activities exactly the same way, between which there are intervals of irregular activity.

3. A scenario where each pattern instance is heavily treated with noise. This correspond to a person that acts with regularity, but is insufficiently instrumented to give trustworthy data, or does the same activity in a different way.

4. A scenario where each pattern is instantiated in varying intervals. A user profile exhibiting this behaviour would be a from a person performing activities identically each time, but not in the same order or at the same frequency.

5. A scenario where 75% of the cells of each pattern are undefined. These sparse patterns represent a faulty sensor suite or someone who only has a few key detectable regularities in their daily activities.

These configurations are summarised in Table 4.1.

A number of other configurations parameters are equal across all configurations and scenarios:

- All patterns have length 10;
- Each simulated sensor has an alphabet size of 10 – a realistic compromise between separation and resolution;
- Transition costs are equal for all configurations;
- The inter-pattern variance is infinite across all configurations – each pattern is generated independently from the others.

Furthermore, we vary three properties in three ways each:

- Number of patterns: 2, 5 or 10;
- Number of rows: 2, 5 or 10;
- Number of columns: 100, 200 or 500 (cf. Table 4.2).

In Table 4.2, advancing a column increments from the base index given in the first column. For example, the 15th configuration can be found in the column denoted +5 in the row denoted 10.

These latter give us the 27 variations of each of the five principal scenarios and thus we obtain the number of 135 configurations. For each scenario, a set of ten patterns is defined, of which suitable subsets are used for each of the configurations.

We define the transition costs for the generator such that the identity transition is assigned a zero cost, whereas a substitution with any other symbol is assigned a cost of one. Substituting a symbol with an insertion or a deletion is given a cost of two (using the classic model of fixed indel scores), and substituting a *don't care* symbol with any alphabet symbol is given a cost of one as well. Table 4.3 contains the entire transition cost matrix.

Table 4.1.: Configuration of the five datasets.

Dataset	1	2	3	4	5
noise interval length (count)	0	10	0	0	0
pattern noise distribution (σ^2)	0	0	9	0	0
pattern repetition distribution (σ^2)	0	0	0	9	0
gappiness (ratio)	0	0	0	0	0.75

Table 4.2.: Configuration values for the 27 experiments making up a dataset.

index	+0	+1	+2	+3	+4	+5	+6	+7	+8	
1	2	5	10	2	5	10	2	5	10	num. of patterns (m)
	2	2	2	5	5	5	10	10	10	num. of rows (n)
	100	100	100	100	100	100	100	100	100	num. of columns (l)
10	2	5	10	2	5	10	2	5	10	num. of patterns (m)
	2	2	2	5	5	5	10	10	10	num. of rows (n)
	200	200	200	200	200	200	200	200	200	num. of columns (l)
19	2	5	10	2	5	10	2	5	10	num. of patterns (m)
	2	2	2	5	5	5	10	10	10	num. of rows (n)
	500	500	500	500	500	500	500	500	500	num. of columns (l)

Table 4.3.: Transition matrix T used for the generation of the test datasets.

	ins	**del**	0	1	2	3	4	5	6	7	8	9	*
*	2	2	1	1	1	1	1	1	1	1	1	1	0
0	2	2	0	1	1	1	1	1	1	1	1	1	1
1	2	2	1	0	1	1	1	1	1	1	1	1	1
2	2	2	1	1	0	1	1	1	1	1	1	1	1
3	2	2	1	1	1	0	1	1	1	1	1	1	1
4	2	2	1	1	1	1	0	1	1	1	1	1	1
5	2	2	1	1	1	1	1	0	1	1	1	1	1
6	2	2	1	1	1	1	1	1	0	1	1	1	1
7	2	2	1	1	1	1	1	1	1	0	1	1	1
8	2	2	1	1	1	1	1	1	1	1	0	1	1
9	2	2	1	1	1	1	1	1	1	1	1	0	1

The similarity score table for the local alignment algorithm is derived from these values. The identity score is set to 10. The substitution of one symbol with a different one is assigned a penalty of -15. This value is chosen as an optimisation for scenario 2, based on the following consideration:

The chance of a random match is 1/10 for each cell. In the two row case, completely bridging the noise interval between two patterns requires an alignment of the 20 cells between two patterns. The expected value of random matches in this interval is two. To improve pattern separation, the penalty value is chosen to prevent over-alignment even for four instances of matches in the interval of noise. This covers 95% of all cases in the two-row configuration.

The insertion and deletion penalty are chosen twice as high (-30) as the substitution penalty, mirroring the generation configuration.

For each one of the 135 configurations, a dataset consisting of ten sequences is generated.

Next, we evaluate our alignment approach on this generated data.

4.1.2 Synthetic Data Pattern Extraction Evaluation

This subsection presents the results obtained from using our local alignment algorithm (as introduced in Chapter 3) on synthetic data generated according to a number of different scenarios. We have generated a total of 135 different datasets, and use the alignment approach to extract patterns, which are then compared to the actual patterns – as generated – in the test data. We decided to limit ourselves to 135 datasets, as we assume this to be a good compromise between covering some of the breadth of possible configurations, and also allowing us to present all the results.

4.1.2.1 Evaluation Criteria

We evaluate the alignments on four criteria.

Number How many alignments are made, with regard to the expected number of possible pairings of patterns between the two sequences?

Precision How much of an alignment actually covers a pattern?

Recall What part of a pattern is covered by an alignment?

Alignment size How big are the alignments that are found, compared to the size of the patterns present?

We average precision and recall across all alignments for a configuration. If an alignment covers multiple patterns, we only consider the best-covered pattern. In Fig. 4.1.7, the precision and recall measures are presented on an example.

4.1.2.2 Hypotheses

With regard to the five scenarios characterising each dataset laid out in the previous section, we can expect the following results:

1. The first dataset, especially in conjunction with low numbers of patterns, should develop "macro-pattern" artefacts (i.e. series of patterns with the same order of individual patterns) and few, but overly large alignments can be expected.

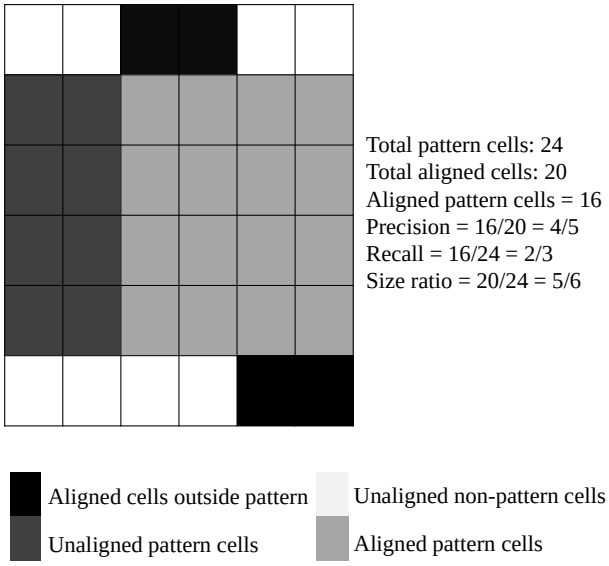

Figure 4.1.7.: Evaluation scoring example.

2. The second dataset should have a low incidence rate for complete over-alignments. These occur when two patterns appear in the same order in two input matrices and the random values between patterns are sufficiently similar. Patterns should otherwise be well discerned. Single-row over-alignments can be expected to be more common (over one third for configurations with only two patterns), especially for larger row numbers.

3. Dataset three can be expected to be a case where the algorithm would not be able to identify most of the patterns. The selected configuration means that almost 80% of all values are changed during instantiation. This makes alignable patterns rare. Despite a minimal admissible score of only 60% compared to datasets one, three and four, complete and correct alignments of patterns are unlikely to be made. The contiguous characteristics of the alignments mean that some changed cells of patterns can be included in the alignments.

4. Results on dataset four should also be similar to those on dataset one, but with – on average – shorter alignments, as macro-patterns should be less likely to emerge, when the order of pattern instantiation is less regular. The impact of this is expected to be more noticeable with configurations with a higher number of patterns. Alignments are more likely to match well with patterns.

5. In the fifth dataset finally, the scoring system in place and the extreme gappiness should have a large percentage of false positives among very few, partial alignments of actual patterns. It can be seen to serve as a negative control experiment. In contrast to set three, the alignments should be more likely to span non-pattern cells.

4.1.2.3 Results

The average evaluation results for each scenario are contained within Table 4.4. This table contains the ratio of detected alignments to expected number of pairs of patterns in the data, the average precision and recall values across all 27 configurations (which are themselves the averages across all alignments for each configuration) and the size ratio. The values for Dataset 3 and Dataset 5 are not directly comparable to the others, as we used a different MASS base score, to account for the noise in Dataset 3 and the missing data in the case of Dataset 5. MASS in these cases have been reduced to 0.6 (a value we consider to be similar to what may pass as a real world noise tolerance value) and 0.25 (three-quarters of the cells of a pattern are undefined) respectively of the corresponding MASS in the other configurations. For these, MASS is equal to the size of the pattern, as we assign a similarity score of one per identical cell.

Table 4.4.: Key results: averages and standard deviations across all configurations.

Measure	Dataset 1	Dataset 2	Dataset 3[1]	Dataset 4	Dataset 5[2]
$\frac{\#\text{alignments}}{\#\text{pairs of pat.}}$	0.31 ± 0.26	0.69 ± 0.29	0.041 ± 0.054	0.31 ± 0.26	0.22 ± 0.32
precision	0.54 ± 0.22	0.77 ± 0.20	0.13 ± 0.06	0.52 ± 0.24	0.54 ± 0.18
recall	1.00 ± 0.00	1.00 ± 0.00	0.56 ± 0.19	1.00 ± 0.01	0.20 ± 0.06
$\frac{\text{alignment size}}{\text{pattern size}}$	5.66 ± 4.48	2.45 ± 2.51	6.72 ± 3.96	6.65 ± 6.08	1.60 ± 0.24

The runtime for the complete set of alignments is around 4 minutes of real time on a pair of Intel® Xeon® E5-2560. Across all 135 configurations, we obtained 214,963 alignments.

In Fig. 4.1.8, we present the ratios of alignments to expected pairs of patterns per configuration. The most striking trend is that datasets 1 and 4, and to a lesser extent dataset 2, show a series of this measure rising in patterns of three. This can be directly attributed to the macro-patterns that are created, which are obviously more numerous when the number of patterns in the data is low, and simultaneously

[1] The results for Sets 3 have been obtained with a MASS of 60% of those of Sets 1,2 and 4.
[2] The results for Sets 5 have been obtained with a MASS of 25% of those of Sets 1,2 and 4.

the lack of noise does not allow segmentation of the patterns into their components. A more subtle trend lies in the reduced number of alignments made, when there are more sensors in the data. This can possibly be ascribed to the higher MASS in use for those scenarios.

Table 4.5.: Average Pattern Size for Dataset One

config.	mean	error
0	4.13	±2.73
1	1.32	±0.93
2	1.82	±1.99
3	1.29	±0.81
4	5.86	±5.50
5	5.25	±5.31
6	2.81	±3.25
7	8.92	±5.61
8	8.99	±5.64
9	4.69	±5.07
10	1.53	±1.27
11	1.23	±0.73
12	3.70	±2.86
13	3.09	±2.53
14	1.89	±1.60
15	4.35	±2.68
16	3.72	±2.75
17	2.98	±2.22
18	8.68	±10.54
19	2.22	±3.11
20	11.26	±12.35
21	8.45	±11.74
22	4.50	±8.19
23	17.55	±14.69
24	17.89	±14.89
25	8.88	±12.30
26	5.83	±4.86

For dataset one, almost all patterns are covered by alignments and average alignment size is 5.66 times the pattern size (validating our macro-pattern hypothesis), two outlier configurations (16 and 17) reach an average pattern size over 17 times larger than patterns, with very large deviations in the samples (cf. detailed results in Annex B.1.2 and an extract in Table 4.5). Mean precision across all configurations is 0.55 (standard deviation across means of each configuration: 0.22).

Dataset two benefits from the fact that it is the reference for the score matrix. This means good separation of patterns due to the noise between them (alignment size on average 2.45 times the pattern size). The average number of alignments is 0.69 times the number of pairs of patterns. Precision is relatively high, and almost all patterns are completely covered by at least one alignment. Problems with over-alignment arise when a low number of patterns is combined with a high number of sensors. Here the score obtained by aligning a pattern with another is high, and the chance that the following patterns match is also high; this means that an alignment stretches over multiple patterns when the score penalty incurred by the noise interval is not sufficient to prevent over-alignment.

Dataset three highlights the detrimental effects of noise on alignment quality and quantity. Especially with the equidistant layout of the value space, there are very few alignments made with the provided score matrix. The preci-

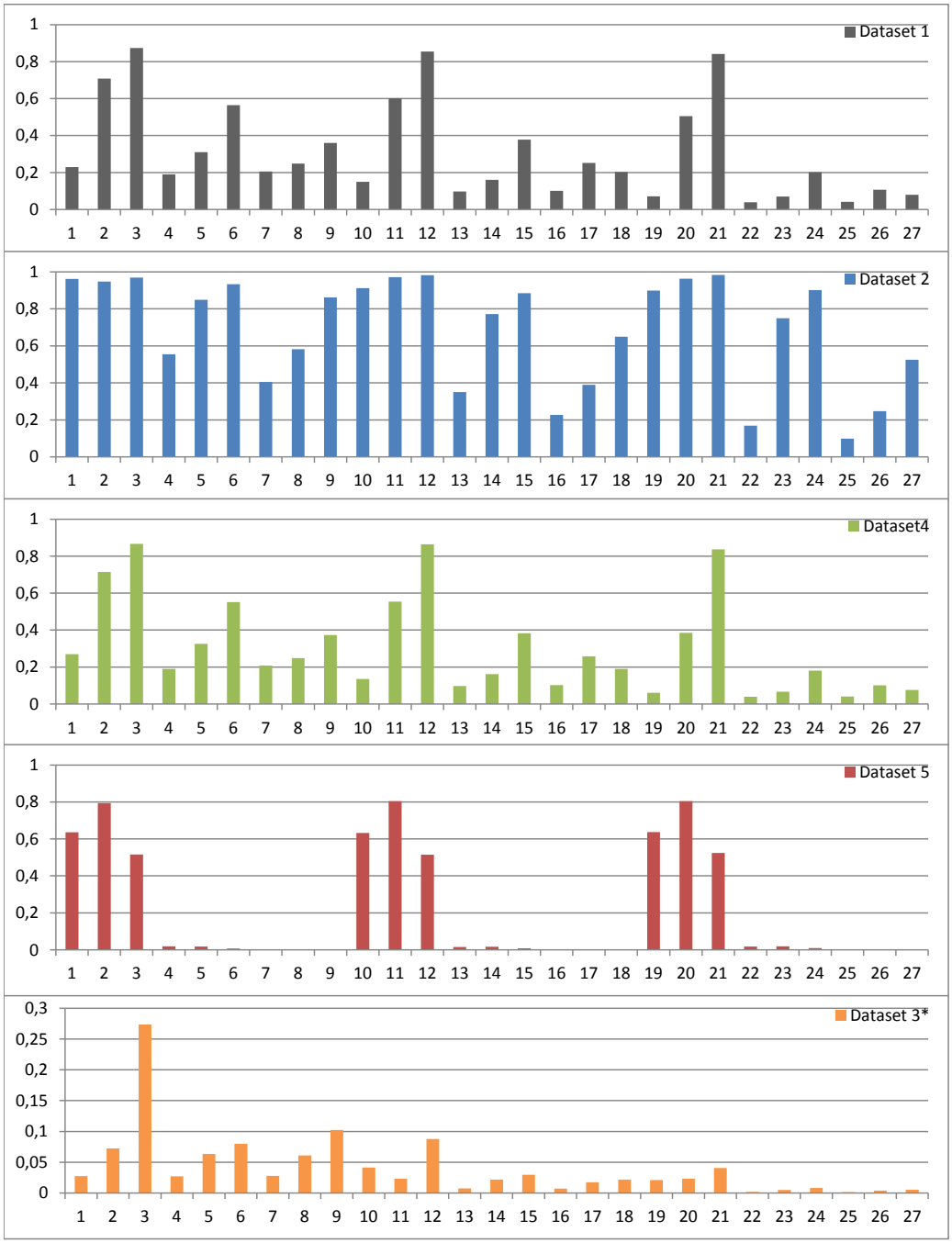

Figure 4.1.8.: The ratios of the number of alignments to the combinatorially expected number of identical pairs of patterns in two different files, for all five datasets and all 27 experiments. As the number of patterns rises, the expected number of pairs gets lower, and the rate of alignments gets closer to the number of pairs.
*Note: The values for dataset 3 are on another scale than the others.

sion score is very low, alignment size varies wildly, but is on average very large, despite a 40% reduction in minimal admissible scores, which should favour smaller alignments. The pattern coverage metric is no longer accurate for this case, as its implementation depends on the simultaneous start of a pattern which is no longer guaranteed once insertions and deletions appear. Notably, some patterns were found in all configurations.

The results from dataset four are in most aspects very similar to those from dataset one. In comparison, the average size of patterns is higher; recall and precision are lower, and show a slight increase in variability.

Dataset five – serving as negative control – has the minimal admissible score of the alignments reduced to one quarter of those of dataset one, to offset the three-quarter loss of information induced by the gappiness value. This visibly does not even out the loss of information due to gappy patterns. The score matrix punishes inequality too heavily for but a few chance alignments to arise, if any at all. Precision is no lower than for datasets one and four, as the low scores of the alignments do not allow the inclusion of a large number of non-matching values in alignments. The average recall for patterns covered by alignments is very low. Alignment sizes reflect the reduced scores and are similar across all configurations (0.05 standard deviation, excluding configurations without alignments).

4.1.2.4 Performance Comparison

Our key contributions having as goal performance improvements of an existing algorithm, we compare these results with those obtained with the original algorithm. Where the previous results – with the new algorithm – were obtained in just over 4 minutes, the original algorithm (as described by Lecroq et al., 2012) gave most results after around 10 hours, but eight configurations with large alignments (particularly configurations of datasets 1 and 3 with long sequence lengths and few different patterns) took over a week to give results, under the same testing configuration.

To verify that this drastic increase in performance did not negatively influence our performance metrics, they are presented in Table 4.6. Even if initially the values appear better (especially precision for datasets one and four) than what we obtained from the faster algorithm, we argue that this is possibly due to an undesired effect. The key difference in results, is that a total of 291,858 alignments are made using the slower approach. Since these alignments can be assumed to be smaller than the ones we obtain using the fast variant (due to the score maximisation effort), they reduce the impact of macro-patterns on the evaluation scores. On the other hand, they are also likely to be redundant, as they should be covered by or similar to another alignment of the same pair

Table 4.6.: Key results: averages and standard deviations across all configurations for the state of the art algorithm

Measure	Dataset 1	Dataset 2	Dataset 3[3]	Dataset 4	Dataset 5[4]
$\frac{\#\text{alignments}}{\#\text{pairs of pat.}}$	0.57 ± 0.27	0.867 ± 0.15	0.022 ± 0.024	0.55 ± 0.27	0.15 ± 0.23
precision	0.70 ± 0.24	0.93 ± 0.06	0.22 ± 0.10	0.69 ± 0.26	0.64 ± 0.21
recall	0.98 ± 0.03	1.00 ± 0.00	0.72 ± 0.32	0.95 ± 0.14	0.20 ± 0.05
$\frac{\text{alignment size}}{\text{pattern size}}$	2.82 ± 2.20	1.12 ± 0.11	6.48 ± 4.21	3.19 ± 3.37	1.40 ± 0.16

of input sequences. The improved average recall values of our new algorithm compared to the older one also are indicative of this. Additionally, our results for the more difficult datasets (three and five) actually show an improvement with regard to the number of alignments found using our new approach over the state of the art algorithm.

We therefore claim that the reduction in results and much decreased time required for the calculation make the resulting reduction in average precision an acceptable trade off, especially because there is evidence that the decrease is due to additional, non-maximal patterns in the data, which are of less interest.

4.1.3 Analysis

Overall, this evaluation shows that the algorithm works largely as expected, and allows us to judge system constraints. Our optimisations on the state of the art algorithm have reduced run time massively. This renders it capable of realistically dealing with data that has a much larger scale than previously possible. On the dataset for which the algorithm was tuned, the results are acceptable, especially when the presence of many patterns inhibits the formation of macro patterns. Recall values in particular indicate that most of the time an alignment corresponds to a pattern.

The results also show the limits of our approach. Although in the ideal case we observe many perfect fits of alignments to patterns, we also observe the tendency of the current configuration to extend alignments beyond pattern boundaries, if identical pattern sequences are present. In sparse data or in noisy data, the simple model we use here prevents reliable extraction of patterns, because all substitutions are equiprobable. This is especially characterised by the low recall values, indicating that most alignments are made from data that is randomly similar to other data.

One of the key observations on these results is that the length of the sequence has a detrimental effect on almost all performance measures. This is due to combinatorial

[3]The results for Sets 3 have been obtained with a MASS of 60% of those of Sets 1,2 and 4.
[4]The results for Sets 5 have been obtained with a MASS of 25% of those of Sets 1,2 and 4.

effects which create macro-patterns in a higher number in these long sequences. Minimising length – so that only a few patterns are included in a sequence – allows for better precision. Especially in real world data, when examining data from people with a routine-heavy lifestyle, one would expect to find larger patterns if the search window is sufficiently large. This is reinforced by the number of different patterns countering this combinatorial effect.

Another observation is that the wider the dataset, the worse the impact of macro-patterns. This is due to the higher accumulated similarity score after each block of identical data, and therefore increased tendency to bridge the gaps between identical blocks. Although this is a problem with ideally replicated pattern instances, in noisy data this increased resiliency might be beneficial. The effect can be controlled by adjusting the similarity scores for the various edit operations.

On the other hand, the actual behaviour on context data can only be determined by testing the approach on real world data. This is explored in the next section.

4.2 Real World Data Evaluation

The previous synthetic data based evaluation of our algorithm allowed us to determine the runtime of our algorithm under specific, controlled conditions, as well as gain some initial insights on how it performs with regard to our accuracy measures. To truly validate our approach, further evaluation on real data is required. In the following we describe how we obtained the real data we use, present our results and then analyse them for potential conclusions.

4.2.1 Data Collection Campaign

The validation of our approach on real world data is a key part of this work. After an examination of available context datasets (cf. section 2.4 on page 46), we come to the conclusion that none of these is suitable for the specific aims of this work. Hence, we designed a specific data collection protocol and launched a data collection campaign in late 2011. This section documents the process of this campaign from conception to an annotated dataset that is in a suitable format for the alignment approach introduced in the previous chapters.

4.2.1.1 Motivation

The key limitations we identified in existing datasets, were the absence of ground truth data and the limited scope of the available context data. Our key goal therefore was to address these shortcomings, by including more physical context data and simultaneously maintaining a reasonable volume of data – so that manual annotation with ground truth data would be possible.

The key design imperative is a small footprint, in the sense of the context data capture mechanism not impeding the daily routine of the test subjects. Additionally, it is important for the data collection mechanism to maximize the amount of data available, both by implementing safeguards to prevent the loss of already recorded data, and by allowing the subjects to selectively disable individual sensors. The latter follows our reasoning that users would be more open to gathering data if they could selectively disable sensors. It is better to lose some data for an interval, than to lose all of it, in case a user wants to disable tracking. This approach – coupled with visible feedback of the current sensor values – aligns our data capture philosophy with the one proposed by Kärkkäinen et al. [2010].

4.2.1.2 Data Collection Campaign

We decided to use an Android application as mobile segment of our data collection software. This application regularly records sensor data and transmits this data in aggregated batches to a storage database server. The following values are being recorded every ten seconds:

- Local time and date,
- Geolocation (via the Android Location API: Based on GPS and radio network fingerprints),
- Linear acceleration forces (3D Accelerometer),
- Angular velocity (3-axis gyroscope),
- Magnetic field (3-axis magnetometer),
- Luminance,
- State of the handset proximity sensor,
- Type of data network currently available,

- Signal strength of cellular network and
- Remaining battery charge.

These values are first written to a local database on each of the phones, and then automatically sent to a central database server every 24 hours, if a wireless LAN connection is available. Otherwise a retransmission is attempted every six hours. A participant in the data collection campaign can also manually trigger a transmission of the locally stored data.

Our campaign was centred around Rouen (a major regional town in northern France) and Grenoble (a university town in south-eastern France). We recruited a total of 20 test subjects, from 5 of which we obtained useful amounts of data. The other test subjects did not participate over the entire term of the campaign, possibly due to lack of incentive or usability issues caused by the increased battery drain of their phones. The campaign ran for a duration of two months, from November 2011 until January 2012. The total amount of data we accumulated was around 430 MB, most of which was from 3 of these 5 users, with the remaining users contributing much less data. Most of this variation is either due to early termination of participation in our campaign, or because many sensors were either disabled by the participant or not available on the participant's hardware.

Besides the obvious privacy concerns of such a campaign, one other issue that presented itself to participants was the high battery usage, particularly of the GPS and micro-electromechanical sensors (e.g. accelerometer, gyroscope, magnetometer). This reduced the battery run time of most of the handsets used in the campaign to significantly less than 24 hours, meaning that recharging the device every night was recommended and necessary.

With regard to the design goals, this was the principal concession we had to make. Using the smart phone as capture platform for the context data, allowed for less disruption in the daily lives than a separate device would have generated. Our other criterion, of capturing a wide variety of physical context data, has been mostly met. Unfortunately many phones lacked gyroscopes and some device specific issues when using the microphone prevented us from reliably capturing environmental noise levels, which ultimately prevented us from including this in our data. Meteorological information also appeared as a promising aspect of context data to us, but at the time of development, thermometers, barometers and hygrometers were not available on phone platforms, and using real time data from a nearby weather station is both unreliable and difficult to implement. Getting this data after the end of the campaign was also an option, but historical data is

only available at low temporal resolutions and from services implementing strict quotas on queries.

4.2.1.3 Data Preparation

The data we collected requires some amount of pre-treatment to fit to the model we have laid out earlier. Although the ten-second acquisition interval gave us a regular time-discrete source of data, most values were still quasi-continuous.

Due to the amount of manual intervention required at this stage, we reduce our test dataset to the data of the test subject which contributed the largest amount of data. Examination of this data reveals that this participant's device lacked a gyroscope, so no angular velocity data is available. Other sensors contain obviously "troublesome" (false readings, singleton outliers) values, which we eliminate. We also discard network type and signal strength data, as they are highly correlated with geolocation. To further reduce the amount of data present – to facilitate manual annotation of patterns – we fuse magnetic and accelerometric orientation data into one single 6-axis sensor. This leaves us with five context data sources:

1. light intensity on a logarithmic scale
2. state of the proximity sensor
3. location
4. battery charge level
5. orientation with regard to the magnetic North Pole and the gravitational centre of the earth.

The next step is the actual discretisation, which is performed manually. Clustering algorithms (such as k-means and X-means [Pelleg and Moore, 2000]) serve as an aid to assign 14 place-IDs to all pairs of longitude and latitude values (see Fig. 4.2.1 for an example), 8 orientation IDs to all 6-tuples of accelerometer and magnetometer readings and 8 discrete battery charge levels. The luxmeter only gives 10 different levels of luminance, which are used directly and the proximity sensor returns a binary reading.

We then reduce the sampling frequency to one measurement every one hundred seconds, and cut the data into 24-hour segments, from 4 a.m. to 4 a.m. the following day. These segments are around 850 5-tuples long, which renders them tractable both for the alignment algorithm (cf. the runtime of our initial experiments in subsection 4.1.2.4) and – to a much larger degree – manual annotation. The latter is a necessary step to allow

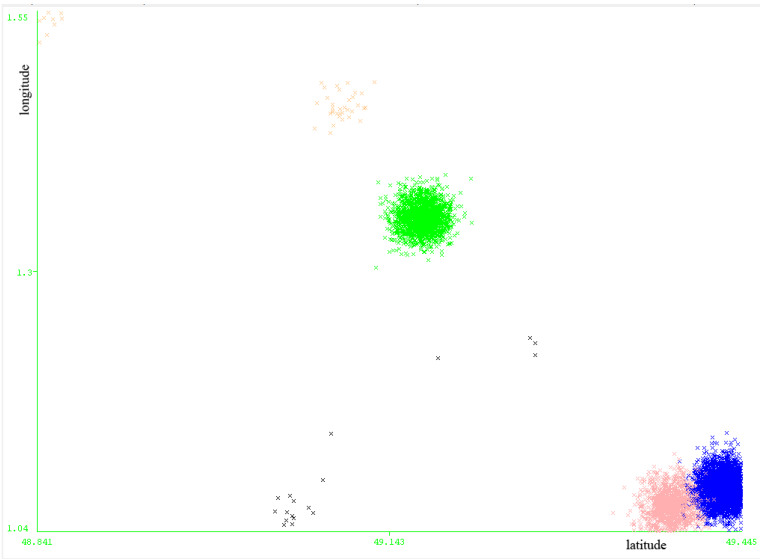

Figure 4.2.1.: Clustering of locations into places of the data gathered by one subject over a one week timespan. Artificial jitter has been introduced to the data, to better visualise data density. Note how nine incidences in the top left are clustered together with the other points towards the top left. The low number of incidences renders this error negligible.

proper evaluation of discovered alignments: Three key patterns (cf. Annex B.1.3) have been selected by visual analysis of graphs of the sensor data (cf. Fig. 4.2.2), and then each occurrence has been labelled by hand, by annotating the raw data with a pattern ID for the sensors and intervals that visually resembled one another.

Although this is an inaccurate practice, it does allow us to roughly label some of the features we expect the algorithm to correctly identify and inter-align. As the choice of "ground truth" is somewhat arbitrary, especially with regard to the vast amount of data present even after the sweeping reductions in data volume, it cannot be understood to be a truly ideal measure. Nonetheless, it is the best measure available under the circumstances, especially considering that the envisioned application is as a decision aid. Being able to detect these key features is a good benchmark of whether this capability of the algorithm also persists for similar data.

4.2.1.4 Post Processing

Once the discretised data is available, the next step is to determine the additional information required to obtain the metadata information (substitution score tables and indel penalties). We assign each pair of states for each sensor a similarity value, using

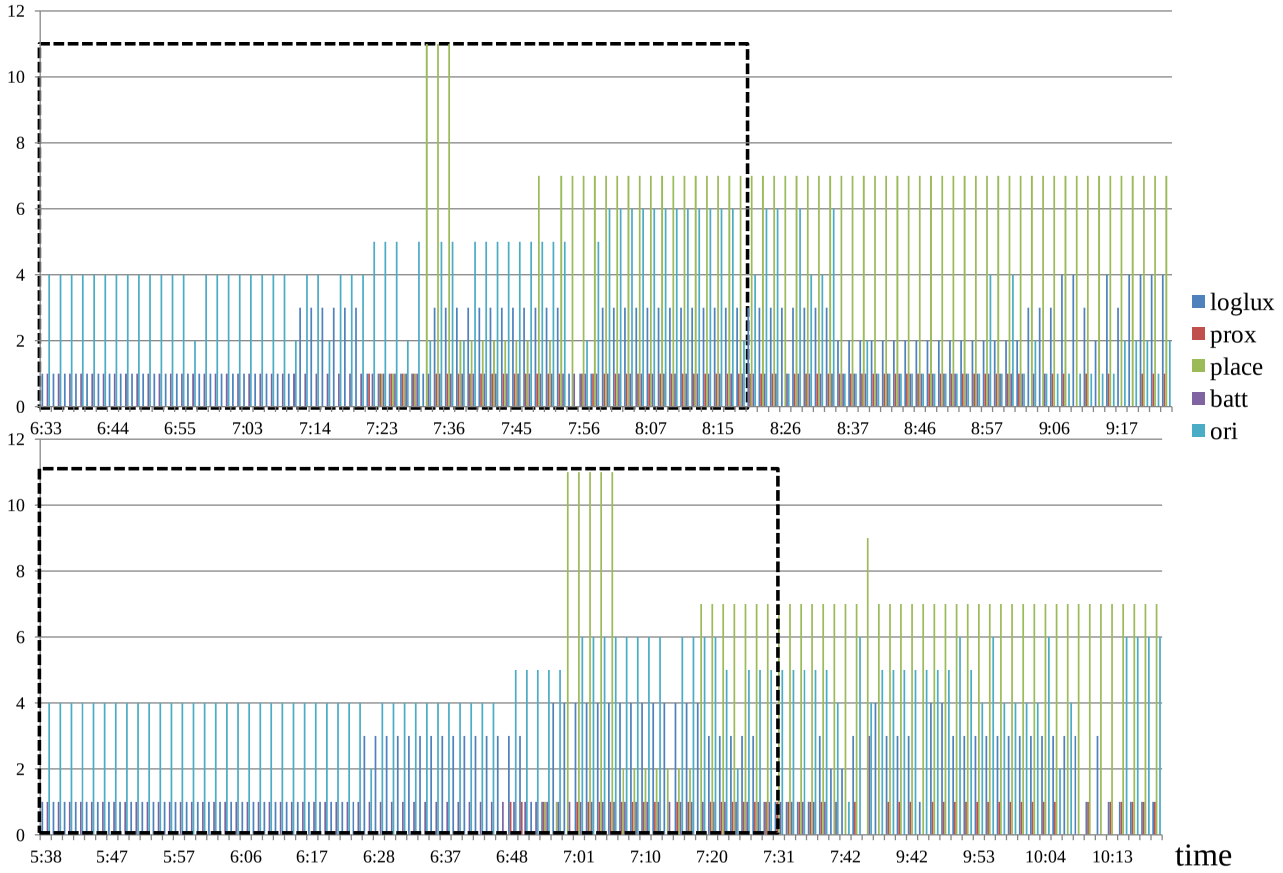

Figure 4.2.2.: Visualisation of an extract of the captured data, with similar intervals marked. This extract covers the morning period of two consecutive days. The y-axis represents the different discretised context values. The values shown in this extract are light intensity on a logarithmic scale (loglux), the state of the proximity sensor (prox), the location cluster id (place), the battery level (batt) and the orientation of the phone (ori).

the approach laid out in the following.

For this evaluation, we create a set of similarity scores derived from both external knowledge of the data sources and transition probabilities between each of the different values. Specifically: In the case of the battery level, we used a linear distance function between the discrete levels, and for everything else we based the similarity measure on the transition probability, and then symmetrised these values. The negative values in the latter case correspond to a projection of the transition probabilities onto a quasi-logarithmic scale, with a score of -10 penalizing less than one transition in 10,000 observations, and a score of -4 corresponding to one transition in five observations. The positive values are around 5 times the transition probability. The special case of the missing reading is dealt with, by assigning a weak malus to any substitution of this value with another and a weak bonus when matched, as we do not consider sensor malfunction and deactivation to be a *reliable* context information, despite the potential significance.

For this data, we expect many temporal extensions and compressions of patterns, and therefore wish to align constant similar sections of different length. Due to the combined insertion/deletion+substitution approach, we set a relatively low negative supplement score for insertion and deletion of -2, compared to the maximum negative score of -10 for transition probabilities smaller than 10^{-5}. The full substitution score tables can be found in Appendix B.1.1.

One key relationship in these tables is that of positive scores to negative scores, as it defines the tolerance of alignments to differences in data. Due to the weakly positive to weakly negative effect of "constant" insertions and deletions, as well as the width of the data we are using and the long periods of constant values in the data, it is preferable to penalize non-identical substitutions heavily. This also shapes more compact alignments, preventing "over-alignment".

With the dataset now laid out, we can use it for the evaluation process.

4.2.2 Evaluation of Alignment Approach on Real Data

This section quantifies how well the alignment approach works to match the manually selected patterns to the instances of these patterns in the dataset. The key parameters are the substitution score tables, as well as the insertion and deletion scores, and finally the minimal accumulated similarity score (MASS, cf. section 3.2). The generation of the tables is described in the previous section, and the choice of MASS is discussed in the following section. Thereupon follow the results we have obtained.

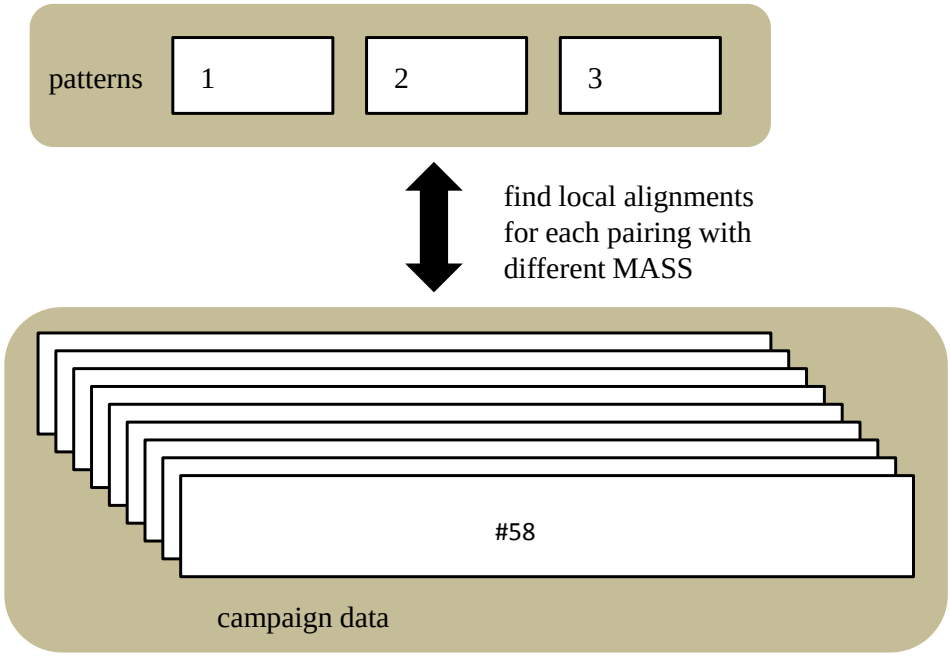

Figure 4.2.3.: Alignment of three pre-selected patterns with campaign data in 58 slices corresponding to 24-hour periods of 5-tuples.

4.2.2.1 Experimental Setup

Besides the model parametrisation, the other parameter that we can control is the minimal accumulated similarity score. This has a key impact in this context, as it affects the minimum size of alignments and the tolerance to differences between two aligned pieces of data. To show the effect of different MASS values, in the following we perform an alignment of the three identified patterns (cf. Annex B.1.3) with all 58 days worth of data from the campaign participant we isolated in the previous subsection, with eight different values for MASS. This process is illustrated in Fig. 4.2.3. The number of manually annotated instances per pattern is as follows:

- Pattern 1: 13 instances,
- Pattern 2: 18 instances,
- Pattern 3: 27 instances.

As a starting point, we examine the three patterns, and expected accumulated similarity scores. The first pattern is 190 5-tuples in size. By analysing the representing sample that is used for searching instances in the remaining data, we can obtain the

score of a perfect match with itself, which gives an upper bound. This score is 2,962 for the first pattern, 3,271 for the second (199 5-tuples) and 1,441 for the third (197 2-tuples – this pattern only covers proximity and location data).

Our choice of a useful lower bound is based on the assumption that an hour of similar values is the minimum scale of interest. As one hour corresponds to 36 5-tuples, and assuming that we require perfect matches during this hour to meet our minimal criterion of similarity, then the target value is around 500, which corresponds to an average similarity row score of ~ 2.78; the average score per row – assuming uniform distribution of all symbols – is ~ 2.65.

Using these two extrema as starting points, we select the following values as our reference points: 500 as lower bound, 700, 850, 1000, as a range that should give expected results for the smaller pattern, and 1,300, 1,700 and 2,200 as a spread more useful for the larger patterns; finally, we choose 3,000 as the upper boundary, which can only be expected to give a single alignment – namely the alignment of the extracted pattern 2 with itself in the corpus.

4.2.2.2 Results

We use precision and recall (i.e. the ratio of cells correctly aligned in an alignment and ratio of cells of an annotated pattern covered by an alignment) for each of the pair of 5-tuple sequences to be aligned, similarly to the evaluation of the synthetic data results. Additionally we determine the number of alignments made, and for each pattern the number of instances we have manually annotated. For the four values of precision and recall, we calculate mean and standard deviation across all alignments for each MASS-configuration. These values can be found in table 4.7.

An alignment consists of two elements: one in the predefined pattern, and one in one of the 58 day-slices of the campaign data. In this table, "A" identifies the part of the alignment in the search sequence, and "B" represents the corresponding aligned part in a sequence from the corpus. Fig. 4.2.4 visualises these results, by plotting recall and precision across each MASS-value and the B-parts of each pattern. When looking at "B" recall values, it is important to note, that alignments of non-annotated data result in a zero value, which then impacts the mean recall and precision for a result. This is reflected in the high standard error for these values.

Table 4.7.: Evaluation results. A is the search pattern, B is the instance in the corpus.

Dataset	Pattern 1				
Measure	prec. A.	rec. A.	prec. B.	rec. B	num.
MASS = 500	1	0.277±0.158	0.024±0.114	0.013±0.083	5011
MASS = 700	1	0.397±0.158	0.072±0.208	0.050±0.171	1049
MASS = 850	1	0.440±0.163	0.100±0.242	0.071±0.200	734
MASS = 1000	1	0.514±0.151	0.138±0.290	0.107±0.245	462
MASS = 1300	1	0.585±0.128	0.201±0.332	0.158±0.285	306
MASS = 1700	1	0.657±0.104	0.329±0.379	0.267±0.336	171
MASS = 2200	1	0.802±0.071	0.712±0.247	0.633±0.288	38
MASS = 3000	X	X	X	X	0[5]
Dataset	Pattern 2				
Measure	prec. A.	rec. A.	prec. B.	rec. B	num.
MASS = 500	1	0.358±0.157	0.045±0.184	0.031±0.129	5201
MASS = 700	1	0.432±0.149	0.070±0.227	0.048±0.160	3206
MASS = 850	1	0.486±0.131	0.078±0.245	0.057±0.176	2240
MASS = 1000	1	0.524±0.125	0.085±0.268	0.064±0.195	1516
MASS = 1300	1	0.566±0.138	0.190±0.385	0.137±0.281	563
MASS = 1700	1	0.605±0.138	0.277±0.442	0.193±0.321	286
MASS = 2200	1	1	1	1	1
MASS = 3000	1	1	1	1	1
Dataset	Pattern 3				
Measure	prec. A.	rec. A.	prec. B.	rec. B	num.
MASS = 500	0.965±0.059	0.574±0.189	0.596±0.387	0.380±0.324	387
MASS = 700	0.955±0.067	0.680±0.146	0.602±0.376	0.357±0.253	248
MASS = 850	0.966±0.054	0.750±0.060	0.744±0.283	0.433±0.209	185
MASS = 1000	0.992±0.014	0.774±0.046	0.820±0.239	0.505±0.216	46
MASS = 1300	1	1	1	1	1
MASS = 1700	X	X	X	X	0[5]
MASS = 2200	X	X	X	X	0[5]
MASS = 3000	X	X	X	X	0[5]

Across all 7,771 alignments made in search for pattern 1 in the corpus, 6,891 alignments are false positives (i.e. with not even partial coverage of the instances). For pattern 2 this is 11,504 out of 13,014 total, and for the third pattern 186 out of 867 alignments do not even partially cover an annotated instance. This allows us to determine an upper bound for the number of partial false positives (880, 1510, 681) – alignments that intersect with annotated pattern instances, consist of subsequences of such instances or are partially

[5]X denotes the absence of data

covered by other alignments. These numbers also include the alignments that best cover a pattern, which can not exceed eight times the number of instances per pattern (104, 144, 216) due to the accumulation of the values across all repetitions of the experiment, and the perfectly matching alignments, which number 20 in total (7, 8, 5).

Even when the MASS is set to 1,700, we still find 205 alignments of pattern 2 within the corpus, that in fact do not correspond to an annotated instance. Some of the excess can be explained by multiple locally optimal alignments, that partially cover a ground-truth annotated interval, other is in sequences that are simply sufficiently similar according to our measure, but in non-annotated parts of the corpus.

The "A" part of the results is less interesting, as the precision values for the first two patterns are necessarily equal to one, due to every element in the search sequence being part of the pattern. In the third pattern, this value indicates how much of the alignment covers the undetermined part of the pattern. Recall values for the first two patterns are directly the ratio of alignment size to pattern size. For the third pattern, this does not hold true, as the alignment can cover non-determined cells.

An additional factor to consider is that some of the alignments might qualify as patterns to an expert, when they are discovered by alignment, despite not having been selected in the non-aided, manual annotation process, due to the fuzzy nature of the latter.

4.2.3 Analysis

The initial observation on these results is that finding pre-identified patterns by means of aligning n-tuples requires extensive pre-analysis by the expert user to formulate a reasonable expectation of MASS and number of pattern instances in the corpus, based on the scoring tables, size of the pattern and variability between pattern instances. Nonetheless, given the reasonable time of execution for our example (6 minutes for the sequential execution of the 24 samples of this experiment, on an Intel® Xeon® E5-2650), it is possible to quickly perform a number of alignment processes with different MASS, and discard those that appear to be too inclusive or too restrictive. If the parameters are chosen fortuitously, then the alignments closely match the expert's expectation of what constitutes a pattern instance, or at the very least point to areas of interest.

We also note that although pattern one and pattern two share almost the same amount of alignments made at the base score of 500, the elimination of false positives with increasing MASS differs greatly between them. The number of alignments for pattern one drops drastically at the step from 500 to 700 and then is reduced more gradually, whereas for pattern two the number of alignments drops more evenly at the lower MASS

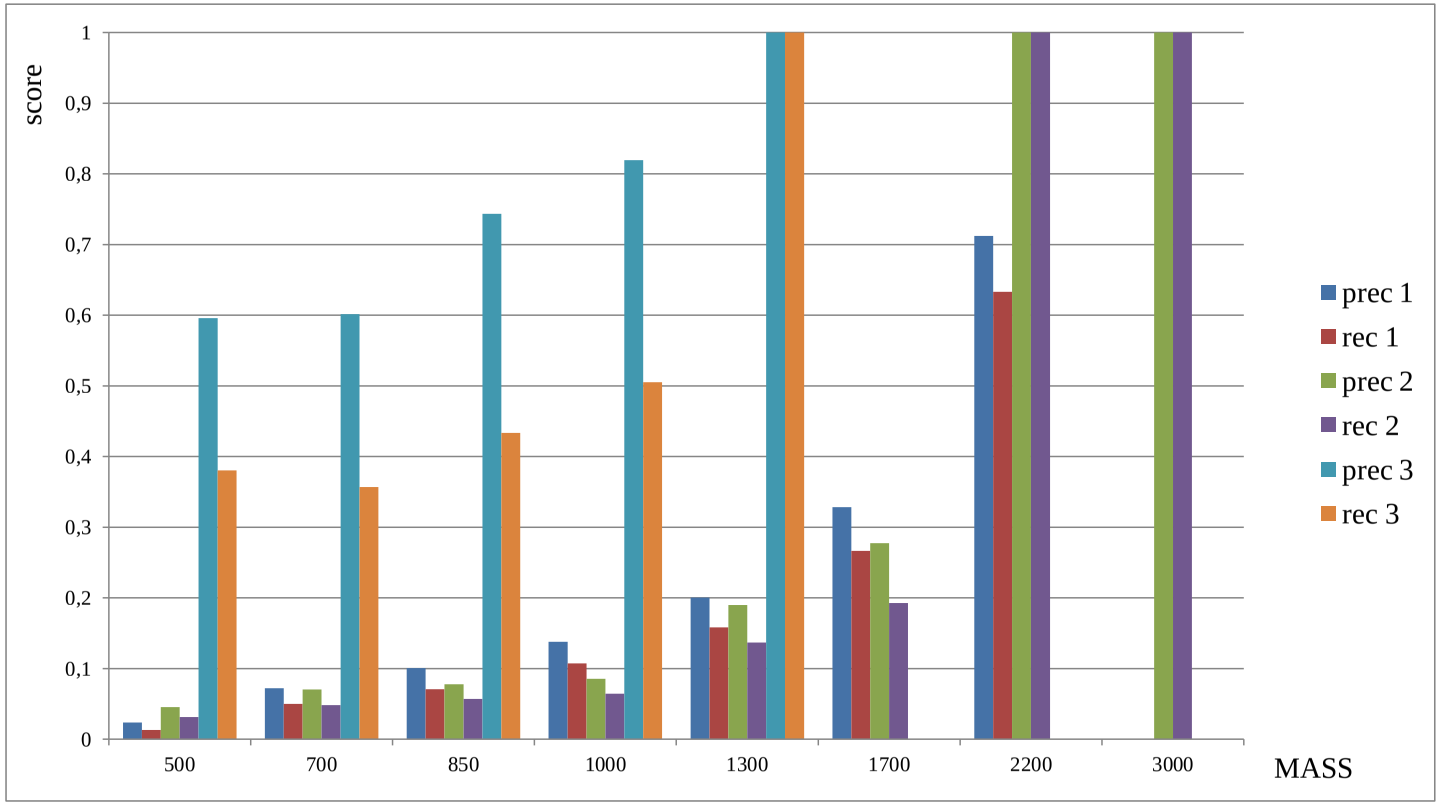

Figure 4.2.4.: Mean precision and recall for the instances in the corpus ("B" - part of the alignment) for the 24 test configurations. From left to right are the different values of MASS used. With rising MASS, both precision and recall increase. The rightmost bars are fewer, as no alignments were made with MASS values 1,700 and above for pattern 3, and no alignments were made with MASS values 3,000 for pattern 1. Due to the smaller number of elements contained in pattern 3, it has higher values than patterns 1 and 2 for the same MASS.

samples, but then very abruptly at a higher score level. And even though the maximum score is higher for pattern two, at MASS 1,700 there are fewer alignments made than for pattern one. This observation suggests that the larger instances of pattern two are also more varied. The consequence is, that as similarity values are accumulated by the alignment algorithm, they stay lower due to constant penalties. Conversely, instances of pattern one have higher local similarity, leading to a better separation of patterns from lower scoring false positives, as penalty scores are fewer and further between. The similarity based approach can therefore also be used to make qualitative observations on sensor data, especially when using a denser sampling of MASS values, for added precision.

Although this evaluation does not cover the alignment of entire day sequences with one another, together with the synthetic evaluation it gives a clear notion of the characteristics of our proposed alignment algorithm, when it is applied on context data. The main challenge to evaluating the algorithm on real data is that this evaluation cannot be done with objective measures, as the notion of what constitutes a "pattern" in context data is not universal, and determining whether an actual routine activity is taking place requires activity diary data in parallel to the data collection campaign, which is unrealistic at relevant scales. Therefore we limit our experimentation to these two aspects.

Conclusions and Future Work

The aim of this work is to determine whether it is possible to detect routine in the day-to-day lives of mobile phone users. We come to the conclusion, that indeed it is possible – if such routine elements are present. We consider this result an extension, refinement and affirmation of existing approaches that detect routine in context data, such as the works of Clarkson [2003] and Eagle and Pentland [2009]. Our results align themselves particularly well with Clarkson's work, in that our approach uses high resolution data, but is only validated against a single person's real world data. What we present is to be seen as a proof of concept, as it lacks additional evaluation against a broader dataset.

Before we continue to discuss the advantages and limitations to our approach, we briefly resume our contribution and results.

We presented in this work an alignment algorithm for sequences of context data, that has been derived from an existing two-dimensional alignment approach, but optimised to take advantage of the characteristics of the data and the desired results, to decrease total calculation time by several orders of magnitude in some cases. To achieve this increased performance, we took into account the structure of context data as a sequence of n-tuples, which allowed us to decrease the original alignment problem complexity by one dimension, and gave a theoretical speed-up of n. Additionally, we used an early discard approach to finding locally optimal alignments in the data, greatly increasing real-wold performance by reducing the amount of backtracking operations required for the dynamic programming algorithm.

This algorithm is then evaluated against two types of datasets, one generated by a specially developed simulator, the other an annotated subset of data recorded during a collection campaign. The results on the former show that our approach works largely as expected, with a weakness when attempting to segment patterns that appear in the same

sequential order in two different input files. On the real world dataset, we used a different evaluation approach to account for the difficulty of obtaining a ground truth reference. The algorithm was used to align known routines with sequences that contained the data of one day each. This result showed that the quality of the results depends directly on the choice of the minimal admissible similarity score, which needs to be optimised to within a small fraction of the ideal value, to obtain results that closely match one's expectations.

One of the main advantages of our approach, is that it can point out from a dataset some candidate intervals that should correspond to pairs of routine elements in context. Our evaluation on synthetic data shows that – if naively configured – the algorithm has a strong tendency to cover multiple patterns, if they appear in the same sequence. It is disputable whether this is always desirable – as it reduces the segmentation of the results – but it follows from our choice to search optimal alignments to reduce the overall number of patterns extracted. An example of where this could be problematic, is when a person frequently does two activities in sequence, but occasionally only one individually. Then – numerically – one routine activity happens more often than the other, but this is not reflected in the result: when aligning two samples that contain the sequence of pairs, the result is a single alignment, with no direct relation to other alignments comprising only one activity.

Our real data evaluation was limited in that we only checked for a set of pre-determined patterns – primarily due to the challenge of manually determining a desirable evaluation target, without a specific target application. Nonetheless, we were able to display how the minimal accumulated similarity score is related to the results. Good accuracy can be expected, if one is capable of expressing one's expectation of what constitutes an alignment in the terms of individual per-element similarity scores and lower accumulated similarity score limits.

In the introduction, we listed a wide variety of applications. When comparing the breadth of requirements different applications may have with regards to models of activity routine with our algorithmic results, we realise that our approach is not an all-in one solution to providing a better understanding of context. This is mostly due to the genericness of our approach, linked with the lack of a common, inherent understanding of what truly makes up a routine activity. What we provide then, is primarily an exploratory tool, enabling domain experts – for example application developers – to gauge across large datasets where there is detectable routine in sensor logs and what the general characteristics of the present routine activities are. They can then iteratively adapt

and parametrise our approach, until they can accurately express their notion of what makes up a routine activity within the design space the alignment paradigm provides. Other potential interest groups include the people generating the data themselves (as a means of introspection and reflection) and researchers in social sciences. Usage in the advertisement industry or intelligence field could also be imagined.

Although such a usage entails some preparation of the context data, the process is relatively straightforward: the first step is synchronisation of the different context streams. Next, the data is discretised, which often requires some manual intervention, to estimate a good number of clusters in the data, or determine a set of criteria to evaluate a clustering result. Based on this step, the meta-data (in the form of substitution score tables) is generated. A statistical approach to this can be mostly automated, a semantic approach requires direct intervention, and may provide better results. In each case, subsequent refinements may be necessary, depending on the first alignment results which reveal some consistency characteristics of the data. This iterative approach can provide a flexible – but complex – tool to extract specifically the patterns a user is interested in. False positives are a frequent occurrence which may have to be rejected by hand. Considering the scale of the raw data, this reduction of the problem of finding routine is a significant step forward, even despite the setup complexity.

We already touched upon three limitations: the iterative, supervised approach is not completely automatable; there may be unexpected alignments among the results, which are classified as false positives; alignments are always maximised and may not extract discrete smaller patterns. Additionally, there are some limitations on the algorithmic level. Currently, our algorithm produces results which are dependent on the order in which the sensor data is arranged; an artefact carried over from the algorithm upon which we developed our adapted approach. We are also faced with a similar "multi-level"-problematic as Clarkson encountered, in that a single accumulated similarity value may be insufficient to characterise all kinds of patterns as alignments. The value may for example be equal for a large alignment with some errors and a small perfect alignment – no qualitative information, besides the number of aligned symbols and maximum similarity score, is retained.

The future work we suggest is threefold. Firstly, there are some ways of modifying the algorithm to improve results and enhance performance. Secondly, there are some means of evaluation that can still be explored to gain an even better understanding of how our approach interacts with context data. Finally, we point to some pre- and post-treatment methods that could render this approach more powerful and simpler to use.

Although we did improve and adapt the algorithm to our use case from a time and space performance standpoint, a weakness still persists: the in-tuple order of elements remains a factor in the alignments found. Correcting this by calculating and memorising the order in which each tuple element is accessed would result in order-independent alignments, at the cost of increased time and space requirements.

A possible solution to the issue of large patterns obscuring smaller ones, can be approached by performing a further pattern extraction step on the results of the first extraction, using a lower MASS. This should reveal whether smaller patterns are present within the larger ones.

The algorithm can be parallelised in its implementation in many ways, which can provide great speed-ups. The one-dimensional string alignment approaches exist in versions optimised for stream processing, porting the ideas of these implementations to our methods could increase performance on specialised hardware, such as graphics processors and processors with streaming extensions. Ultimately, the memory limitation will persist.

We currently lack evaluation of actual pattern extraction on real world data, due the reasons we stated earlier in this chapter. A possible way to evaluate our approach, as a decision aid, would be to undertake user studies, to see how potential users – interested in finding patterns in context data – would be able to parametrise a model that enables them to do just that.

Furthermore, a comparative evaluation against the approach of Plantevit et al. [2010] would be of interest, to see how the ultimate parametrisation efforts compare, as well as to be able to judge the results against a similar approach.

Lastly, evaluating the extraction performance against a dataset supported with diary entries would be able to provide further insights – but ultimately be limited by the flexible notion of what makes up a routine activity. Each test subject providing a diary may have a different notion of routine from the other test subjects, or the experimenter ultimately constructing a model, which can lead to data that is nearly impossible to correctly interpret. As a result, either a naive model would be used, with results probably similar to those we obtained on synthetic data, or a properly developed, supervised model, which would again only demonstrate how well the model can be parametrised to correspond to a consistent but arbitrary notion of routine.

We have already proposed some pre-treatment methods, to obtain a model that is partly automatically generated from a corpus of known data. Using advanced clustering algorithms, and defining a set of heuristics for the permissible error rates for specific sensors, as well as filtering rules, would further simplify the pre-treatment. From a post-

treatment perspective, it is interesting to reduce the vast amount of pairs of results into representatives of actual routine activities. This has been done in the work of Pauchet et al. [2013], using a clustering approach. This approach can be extended, by using a variation of Hirschberg's algorithm [Hirschberg, 1975] to obtain a local cumulative edit distance function across the different aligned elements. This would allow us to cluster multiple inter-similar context episodes to clusters of routine contexts, while also finding similar subsegments. Alternatively, a global alignment can be used, if the further segmentation has already been performed separately.

Bibliography

Mohamed Abouelhoda and Moustafa Ghanem. String Mining in Bioinformatics. In Mohamed Medhat Gaber, editor, *Scientific Data Mining and Knowledge Discovery: Principles and Foundations*, pages 207–247. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-02787-1. doi: 10.1007/978-3-642-02788-8. URL `http://www.springerlink.com/index/10.1007/978-3-642-02788-8`.

Ramesh C. Agarwal, Charu C. Aggarwal, and V. V. V. Prasad. A Tree Projection Algorithm for Generation of Frequent Item Sets. *Journal of Parallel and Distributed Computing*, 61(3):350–371, March 2001. ISSN 0743-7315. doi: 10.1006/jpdc.2000.1693. URL `http://www.sciencedirect.com/science/article/pii/S0743731500916939`.

Rakesh Agrawal and Ramakrishnan Srikant. Fast Algorithms for Mining Association Rules. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *Int'l Conf. Very Large Databases (VLDB '94)*, pages 487–499, Santiago de Chile, Chile, 1994. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 3–14. IEEE Comput. Soc. Press, 1995. ISBN 0-8186-6910-1. doi: 10.1109/ICDE.1995.380415.

Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216, Washington, D.C., 1993. ACM New York, NY, USA.

Stephen F. Altschul and Bruce W Erickson. Optimal sequence alignment using af-

fine gap costs. *Bulletin of Mathematical Biology*, 48(5-6):603–616, 1986. ISSN 0092-8240. doi: 10.1016/S0092-8240(86)90010-8. URL http://www.sciencedirect.com/science/article/pii/S0092824086900108.

Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990. URL http://www.cmu.edu/bio/education/courses/03510/LectureNotes/Altschul1990.pdf.

Amihood Amir and Martin Farach. Efficient 2-dimensional approximate matching of non-rectangular figures. In *SODA '91 Proceedings of the second annual ACM-SIAM symposium on Discrete algorithms*, number 908, pages 212–223. Society for Industrial and Applied Mathematics Philadelphia, PA, USA, 1991. URL http://dl.acm.org/citation.cfm?id=127829.

Sarabjot Singh Anand and Bamshad Mobasher. Contextual Recommendation. In Bettina Berendt, Andreas Hotho, Dunja Mladenic, and Giovanni Semeraro, editors, *From Web to Social Web Discovering and Deploying User and Content Profiles*, volume 4737 of *Lecture Notes in Computer Science*, chapter 8, pages 142–160. Springer Berlin Heidelberg, 2007. ISBN 9783540749509. doi: 10.1007/978-3-540-74951-6_8. URL http://www.springerlink.com/content/r28874294253q051.

Alberto Apostolico, Laxmi Parida, and Simona E. Rombo. Motif patterns in 2D. *Theoretical Computer Science*, 390(1):40–55, January 2008. ISSN 0304-3975. URL http://www.sciencedirect.com/science/article/pii/S0304397507007645.

Daniel Ashbrook and Thad Starner. Learning significant locations and predicting user movement with GPS. In *Proceedings of the 6th IEEE International Symposium on Wearable Computers*, pages 101–108, Seattle, WA, USA, 2002. IEEE Computer Society, Washington, DC, USA. ISBN 0-7695-1816-8. doi: 10.1109/ISWC.2002.1167224. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1167224.

Jay Ayres, Jason Flannick, Johannes Gehrke, and Tomi Yiu. Sequential Pattern mining using a bitmap representation. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '02*, page 429, New York, New York, USA, 2002. ACM Press. ISBN 158113567X. doi: 10.1145/775107.775109. URL http://portal.acm.org/citation.cfm?doid=775047.775109.

Ricardo A. Baeza-Yates and Gonzalo Navarro. Fast two-dimensional approximate pattern matching. In Claudio L. Lucchsei and Arnaldo V. Moura, editors, *LATIN'98 Proceedings of the Third Latin American Symposium on Theoretical Informatics*, pages 341–351, Campinas, Brazil, 1998. Springer-Verlag London, UK. URL `http://link.springer.com/chapter/10.1007/BFb0054334`.

Brenda S. Baker. A program for identifying duplicated code. In *Computing Science and Statistics*, pages 24:49–57, College Station, TX, USA, 1992.

Theodore P. Baker. A Technique for Extending Rapid Exact-Match String Matching to Arrays of More than One Dimension. *SIAM Journal on Computing*, 7(4):533–541, November 1978. ISSN 0097-5397. doi: 10.1137/0207043. URL `http://dx.doi.org/10.1137/0207043`.

Matthias Baldauf, Schahram Dustdar, and Florian Rosenberg. A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, 2(4): 263–277, 2007.

Leonard E. Baum, Ted Petrie, George Soules, and Norman Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The annals of mathematical statistics*, 41(1):164–171, 1970.

Victoria Bellotti, Bo Begole, Ed H. Chi, Nicolas Ducheneaut, Ji Fang, Ellen Isaacs, Tracy King, Mark W. Newman, Kurt Partridge, Bob Price, Paul Rasmussen, Michael Roberts, Diane J. Schiano, and Alan Walendowski. Activity-based serendipitous recommendations with the Magitti mobile leisure guide. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1157–1166, New York, NY, USA, 2008. ACM. ISBN 9781605580111. doi: 10.1145/1357054.1357237.

Martin Berchtold, Matthias Budde, Hedda R. Schmidtke, and Michael Beigl. An extensible modular recognition concept that makes activity recognition practical. In Rüdiger Dillmann, Jürgen Beyerer, Uwe Hanebeck, and Tanja Schultz, editors, *Proceedings of the 33rd annual German conference on Advances in artificial intelligence*, pages 400–409. Springer-Verlag Berlin, Heidelberg, 2010. ISBN 3-642-16110-3 978-3-642-16110-0. doi: 10.1007/978-3-642-16111-7_46.

Kevin Beyer and Raghu Ramakrishnan. Bottom-up computation of sparse and Iceberg CUBEs. In *SIGMOD '99 Proceedings of the 1999 ACM SIGMOD international con-*

ference on Management of data, pages 359–370, Philadelphia, PA, 1999. ACM New York, NY, USA.

V Boonjing and P Songram. Efficient Algorithms for Mining Closed Multidimensional Sequential Patterns, 2007.

A. J. Bernheim Brush, Amy K. Karlson, James Scott, Raman Sarin, Andy Jacobs, Barry Bond, Oscar Murillo, Galen Hunt, Mike Sinclair, Kerry Hammil, and Steven Levi. User experiences with activity-based navigation on mobile devices. In *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, pages 73–82, 2010. doi: 10.1145/1851600.1851616. URL `http://portal.acm.org/citation.cfm?id=1851616`.

Matthew Chalmers. A Historical View of Context. *Computer Supported Co-operative Work (CSCW)*, 13(3-4):223–247, August 2004. ISSN 0925-9724. doi: 10.1007/s10606-004-2802-8. URL `http://www.springerlink.com/index/10.1007/s10606-004-2802-8`.

Tanzeem Choudhury, Gaetano Borriello, Sunny Consolvo, Dirk Haehnel, Beverly Harrison, Bruce Hemingway, Jeffrey Hightower, Predrag "Pedja" Klasnja, Karl Koscher, Anthony Lamarca, James A. Landay, Louis LeGrand, Jonathan Lester, Ali Rahimi, Adam Rea, and Denny Wyatt. The mobile sensing platform: An embedded activity recognition system. *Pervasive Computing*, 7(2):32–41, 2008.

Brian Clarkson and Alex (Sandy) Pentland. Unsupervised clustering of ambulatory audio and video. In *Proceedings of the 1999 International Conference on Acoustics, Speech, and Signal Processing*, pages 3037–3040 vol. 6, Phoenix, Arizona, United States, 1999. IEEE. doi: 10.1109/ICASSP.1999.757481. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=757481`.

Brian Patrick Clarkson. *Life Patterns : structure from wearable sensors*. PhD thesis, MIT, 2003.

Diane J. Cook, Michael Youngblood, Edwin O. Heierman, Karthik Gopalratnam, Sira Rao, Andrey Litvin, and Farhan Khawaja. MavHome : An Agent-Based Smart Home. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications (PerCom 2003)*, pages 521–5244, Fort Worth, TX, 2003. IEEE. ISBN 0769518931.

Brian A. Davey and Hilary A. Priestley. *Introduction to Lattices and Order (2. ed.)*. Cambridge University Press, 2002. ISBN 978-0-521-78451-1.

Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, pages 71–80, New York, NY, USA, 2000. ACM. ISBN 1-58113-233-6. doi: 10.1145/347090.347107. URL http://doi.acm.org/10.1145/347090.347107.

Paul Dourish. What we talk about when we talk about context. *Personal and Ubiquitous Computing*, 8(1):19–30, February 2004. ISSN 1617-4909. doi: 10.1007/s00779-003-0253-8. URL http://www.springerlink.com/openurl.asp?genre=article&id=doi:10.1007/s00779-003-0253-8.

Nathan Eagle and Alex (Sandy) Pentland. Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing*, 10(4):255–268, November 2005. ISSN 1617-4909. doi: 10.1007/s00779-005-0046-3. URL http://link.springer.com/10.1007/s00779-005-0046-3.

Nathan Eagle and Alex (Sandy) Pentland. Eigenbehaviors: identifying structure in routine. *Behavioral Ecology and Sociobiology*, 63(7):1057–1066, April 2009. ISSN 0340-5443. doi: 10.1007/s00265-009-0739-0. URL http://www.springerlink.com/index/10.1007/s00265-009-0739-0.

Vincent Etter, Mohamed Kafsi, and Ehzan Kazemi. Been There, Done That : What Your Mobility Traces Reveal about Your Behavior. In *Nokia Mobile Data Challenge - Next Place Prediction*, 2012.

Simone Faro and Thierry Lecroq. The Exact Online String Matching Problem : a Review of the Most Recent Results. *ACM Computing Surveys (CSUR)*, 45(2):Article No. 13, 2013. doi: 10.1145/0000000.0000000.

Huiji Gao, Jiliang Tang, and Huan Liu. Mobile Location Prediction in Spatio-Temporal Context. In *Nokia Mobile Data Challenge - Next Place Prediction*, number 2, 2012.

Fosca Giannotti, Mirco Nanni, and Dino Pedreschi. Efficient mining of temporally annotated sequences. In *In Proc. SDM'06*, pages 348–359, 2006.

Fosca Giannotti, Mirco Nanni, Dino Pedreschi, and Fabio Pinelli. Trajectory pattern mining. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 330–339, San Jose, CA, 2007. ISBN 9781595936097.

Marta C. Gonzalez, Cesar A. Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, June 2008. ISSN 0028-0836.

Gösta Grahne and Jianfei Zhu. Efficiently Using Prefix-trees in Mining Frequent Itemsets. In *Proceedings of the ICDM'03 international workshop on requent itemset mining implementations (FIMI '03)*, volume 15, pages 123–132, Melbourne, FL, 2003. URL `http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-90/grahne.pdf`.

M A Hall. *Correlation-based feature selection for machine learning*. PhD thesis, The University of Waikato, 1999.

Mark Hall, Eibe Frank, Geoffrey Hilmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1), 2009.

Jiawei Han and Yongjian Fu. Discovery of multiple-level association rules from large databases. In *Proceeding of the 21st international conference on very large data bases (VLDB'95)*, pages 420–431, Zurich, Switzerland, 1995.

Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. FreeSpan: frequent pattern-projected sequential pattern mining. In *KDD '00 Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 355–259. ACM New York, NY, USA, 2000a.

Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *SIGMOD '00 Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 1–12, Dallas, TX, 2000b. ISBN 1581132182. doi: 10.1145/335191.335372.

Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining: concepts and techniques*. Morgan kaufmann, 2006. ISBN 0080475582.

Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15(1):55–86, January 2007. ISSN 1384-5810. doi: 10.1007/s10618-006-0059-1. URL `http://link.springer.com/10.1007/s10618-006-0059-1`.

Martin Heidegger. *Sein und Zeit*. Max Niemayer Verlag, Tübingen, 10 (1963) edition, 1927.

Steven Henikoff and Jorja G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the United States of America*, 89(22):10915–10919, November 1992. ISSN 0027-8424.

Johan Himberg, Kalle Korpiaho, Heikki Mannila, Johanna Tikanmäki, and Hannu T. T. Toivonen. Time series segmentation for context recognition in mobile devices. In *Proceedings 2001 IEEE International Conference on Data Mining*, volume c, pages 203–210. IEEE Comput. Soc, 2001. ISBN 0-7695-1119-8. doi: 10.1109/ICDM.2001.989520. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=989520`.

Daniel S. Hirschberg. A Linear Space Algorithm for Computing Maximal Common Subsequences. *Communications of the ACM*, 18(6):341–343, 1975.

Wassily Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963. ISSN 0162-1459. doi: 10.1080/01621459.1963.10500830. URL `http://amstat.tandfonline.com/doi/abs/10.1080/01621459.1963.10500830`.

Geoffrey Holmes, Bernhard Pfahringer, Richard Kirkby, Eibe Frank, and Mark Hall. Multiclass alternating decision trees. In *ECML*, pages 161–172. Springer, 2001.

Geoffrey Holmes, Bernhard Pfahringer, Richard Kirkby, Eibe Frank, and Mark Hall. Multiclass Alternating Decision Trees. In *Proceedings of the 13th European Conference on Machine Learning*, ECML '02, pages 161–172, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-44036-4. URL `http://dl.acm.org/citation.cfm?id=645329.650070`.

George H. John and Pat Langley. Estimating Continuous Distributions in Bayesian Classifiers. In *Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345, San Mateo, 1995. Morgan Kaufmann.

Juyoung Kang and Hwan-Seung Yong. Mining spatio-temporal patterns in trajectory data. *Journal of Information Processing Systems*, 6(4):521–536, 2010.

Tuula Kärkkäinen, Tuomas Vaittinen, and Kaisa Väänänen-Vainio-Mattila. I Don't Mind Being Logged, but Want to Remain in Control: A Field Study of Mobile Activity and Context Logging. In *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*, pages 163–172, Atlanta, GA, USA, 2010. ACM New York, NY, USA. ISBN 9781605589299.

Richard M Karp and M O Rabin. Efficient randomized pattern-matching algorithms, 1987.

Dimitrios Katsaros and Yannis Manolopoulos. A Suffix Tree Based Prediction Scheme for Pervasive Computing Environments. In Panayiotis Bozanis and Elias N. Houstis, editors, *10th Panhellenic Conference on Informatics, PCI 2005*, pages 267–277, Volos, Greece, 2005.

Dimitrios Katsaros, Alexandros Nanopoulos, Murat Karakaya, Gokhan Yavas, Ozg ur Ulusoy, and Yannis Manolopoulos. Clustering mobile trajectories for resource allocation in mobile environments. In *Proceedings of the 5th International Symposium on Intelligent Data Analysis, IDA 2003*, number 102, pages 319–329, Berlin, Germany, 2003. Springer Berlin Heidelberg.

Eamonn Keogh, Jessica Lin, and Wagner Truppel. Clustering of time-series subsequences is meaningless: implications for previous and future research. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM'03)*, pages 115–122, 2003. ISBN 0769519784.

Niko Kiukkonen, Jan Blom, Olivier Dousse, Daniel Gatica-Perez, and Juha K. Laurila. Towards rich mobile phone datasets: Lausanne data collection campaign. In *Proceedings of ACM international conference on pervasive services (ICPS)*, Berlin, 2010.

Donald E. Knuth, James H. Morris, Jr, and Vaughan R. Pratt. Fast pattern matching in strings. *SIAM journal on computing*, 6(2):323–350, 1977. URL `http://epubs.siam.org/doi/abs/10.1137/0206024`.

David Kotz and Kobby Essien. Analysis of a campus-wide wireless network. *Wireless Networks*, 11(1-2):115–133, 2005. ISSN 1022-0038.

Kamala Krithivasan and R Sitalakshmi. Efficient two-dimensional pattern matching in the presence of errors. *Information Sciences*, 43(3):169–184, December 1987. ISSN 0020-0255. doi: 10.1016/0020-0255(87)90037-5. URL `http://www.sciencedirect.com/science/article/pii/0020025587900375`.

Hye-Chun Kum, Jian Pei, Wei Wang, and Dean Duncan. ApproxMAP: Approximate mining of consensus sequential patterns. In *Third SIAM International Conference on Data Mining (SIAM-DM)*, pages 311–315, San Francisco, CA, 2003.

Kari Laasonen. Clustering and prediction of mobile user routes from cellular data. In *Knowledge Discovery in Databases: PKDD 2005*, pages 569–576, Porto, Portugal, 2005.

Philip Laird. Identifying and using patterns in sequential data. In KlausP. Jantke, Shigenobu Kobayashi, Etsuji Tomita, and Takashi Yokomori, editors, *Algorithmic Learning Theory SE - 1*, volume 744 of *Lecture Notes in Computer Science*, pages 1–18. Springer Berlin Heidelberg, 1993. ISBN 978-3-540-57370-8. doi: 10.1007/3-540-57370-4_33.

Juha K. Laurila, Daniel Gatica-Perez, Imad Aad, Jan Blom, Olivier Bornet, Trinh-Minh-Tri Do, Olivier Dousse, Julien Eberle, and Markus Miettinen. The mobile data challenge: Big data for mobile computing research. In *Mobile Data Challenge by Nokia Workshop in Conjunction with Int. Conf. on Pervasive Computing*, Newcastle, UK, 2012.

Thierry Lecroq, Alexandre Pauchet, Émilie Chanoni, and Gerardo Solano Ayala. Pattern discovery in annotated dialogues using dynamic programming. *International Journal of Intelligent Information and Database Systems*, 6(6):603–618, 2012.

Philip I. S. Lei and Angus K. Y. Wong. The Multiple-Touch User Interface Revolution. *IT Professional*, 11(February):42–49, 2009.

Jonathan Lester, Tanzeem Choudhury, and Gaetano Borriello. A practical approach to recognizing physical activities. *Pervasive Computing*, pages 1–16, 2006. doi: 10.1.1.138.6972.

Chao Li and Katharine Willis. Modeling context aware interaction for wayfinding using mobile devices. In *Proceedings of the 8th conference on Human-computer interaction with mobile devices and services - MobileHCI '06*, page 97, New York, New York, USA, 2006. ACM. ISBN 1595933905. doi: 10.1145/1152215.1152235. URL `http://portal.acm.org/citation.cfm?doid=1152215.1152235`.

David J. Lipman and William R. Pearson. Rapid and sensitive protein similarity searches. *Science*, 227(4693):1435–1441, March 1985. doi: 10.1126/science.2983426. URL `http://www.sciencemag.org/content/227/4693/1435.abstract`.

H Liu and R Setiono. A probabilistic approach to feature selection - A filter solution. In *13th International Conference on Machine Learning*, pages 319–327, 1996.

Nizar R. Mabroukeh and C. I. Ezeife. A taxonomy of sequential pattern mining algorithms. *ACM Computing Surveys*, 43(1):3:1–3:41, November 2010. ISSN 03600300. doi: 10.1145/1824795.1824798. URL `http://portal.acm.org/citation.cfm?doid=1824795.1824798`.

Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Efficient Algorithms for Discovering Association Rules. In *AAAI Workshop on Knowledge Discovery in Databases (KDD-94)*, number July, pages 181–192. AAAI Press, 1994.

Jani Mäntyjärvi, Johan Himberg, Petri Kangas, Urpo Tuomela, and Pertti Huuskonen. Sensor Signal Data Set for Exploring Context Recognition of Mobile Devices. In *Workshop "Benchmarks and a database for context recognition" in conjuction with the 2nd Int. Conf. on Pervasive Computing (PERVASIVE 2004)*, Linz/Vienna, Austria, 2004.

Edward M. McCreight. A space-economical suffix tree construction algorithm. *Journal of the ACM (JACM)*, 23(2):262–272, 1976. URL `http://dl.acm.org/citation.cfm?id=321946`.

Carl H. Mooney and John F. Roddick. Sequential Pattern Mining - Approaches and Algorithms. *ACM Computing Surveys (CSUR)*, 45(2):19:1–19:39, 2013. doi: 10.1145/2431211.2431218.

Cory S. Myers and Lawrence R. Rabiner. A Comparative Study of Several Dynamic Time-Warping Algorithms for Connected-Word. *Bell System Technical Journal*, 60 (7):1389–1409, 1981.

Eugene W. Myers and Webb Miller. Optimal alignments in linear space. *Computer applications in the biosciences : CABIOS*, 4(1):11–17, March 1988. doi: 10.1093/bioinformatics/4.1.11. URL `http://bioinformatics.oxfordjournals.org/content/4/1/11.abstract`.

Mirco Nanni, Roberto Trasarti, Chiara Renso, Fosca Giannotti, and Dino Pedreschi. Advanced Knowledge Discovery on Movement Data with the GeoPKDD system. In *Proceedings of the 13th International Conference on Extending Database Technology EDBT '10*, pages 693–696. ACM New York, NY, USA, 2010. ISBN 9781605589459.

Gonzalo Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, March 2001. ISSN 03600300. doi: 10.1145/375360.375365. URL `http://portal.acm.org/citation.cfm?doid=375360.375365`.

Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

Kurt Partridge and Bob Price. Enhancing Mobile Recommender Systems with Activity Inference. *User Modeling, Adaptation, and Personalization*, pages 307–318, 2009.

Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering Frequent Closed Itemsets for Association Rules. In Catriel Beeri and Peter Buneman, editors, *Database Theory – ICDT '99 SE - 25*, volume 1540 of *Lecture Notes in Computer Science*, pages 398–416. Springer Berlin Heidelberg, 1999. ISBN 978-3-540-65452-0. doi: 10.1007/3-540-49257-7_25. URL `http://dx.doi.org/10.1007/3-540-49257-7_25`.

Alexandre Pauchet, Abed Mohamad El, Tayeb Merabti, Élise Prieur, Thierry Lecroq, and Stéfan Darmoni. Identification de répétitions dans les navigations au sein d'un catalogue de santé. *Revue d Intelligence Artificielle*, 23(1):113–132, 2009. URL `http://hal.archives-ouvertes.fr/hal-00450114`.

Alexandre Pauchet, François Rioult, Émilie Chanoni, Zacharie Ales, and Ovidiou Serban. Advances on Dialogue Modelling Interactive Narration Requires Prominent Interaction and Emotion. In *International Conference on Agents and Artificial Intelligence*, pages 527–530, Barcelona, Spain, 2013.

Jian Pei, Jiawei Han, Behzad Mortazavi-asl, and Hua Zhu. Mining Access Patterns Efficiently from Web Logs. In *Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2000*, volume 0, pages 396–407, Kyoto, Japan, 2000. Springer Berlin Heidelberg.

Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceeding of the 2001 international conference on data engineering (ICDE'01)*, pages 215–224, Heidelberg, Germany, 2001.

Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Jianyong Wang, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Mining Sequential Patterns by Pattern-Growth : The PrefixSpan Approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(10):1424–1440, 2004.

Dan Pelleg and Andrew Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 727–734, 2000.

Helen Pinto, Jiawei Han, Jian Pei, Ke Wang, Qiming Chen, and Umeshwar Dayal. Multi-dimensional sequential pattern mining. In *Proceedings of the tenth international*

conference on Information and knowledge management - CIKM'01, pages 81–88, New York, New York, USA, 2001. ACM. ISBN 1581134363. doi: 10.1145/502598.502600.

James Pitkow and Peter Pirolli. Mining longest repeating subsequences to predict world wide web surfing. In *Proceedings of USITS '99: The 2nd USENIX Symposium on Internet Technologies & Systems*, pages 139–150, 1999.

Marc Plantevit, Anne Laurent, Dominique Laurent, Maguelonne Teisseire, and Yeow Wei Choong. Mining multidimensional and multilevel sequential patterns. *ACM Transactions on Knowledge Discovery from Data*, 4:4:0–4:37, 2010. doi: 10.1145/1644873.1644877. URL `http://dl.acm.org/citation.cfm?id=1644877`.

John C. Platt. Advances in kernel methods. chapter Fast train, pages 185–208. MIT Press, Cambridge, MA, USA, 1999. ISBN 0-262-19416-3. URL `http://dl.acm.org/citation.cfm?id=299094.299105`.

Mika Raento. Mobile communication and context dataset. In *Proceedings of the Workshop towards Benchmarks and a Database for Context Recognition, International Conference on Pervasive Computing*, Vienna, Austria, 2004.

Mika Raento, Antti Oulasvirta, Renaud Petit, and Hannu Toivonen. ContextPhone: A Prototyping Platform for Context-Aware Mobile Applications. *IEEE Pervasive Computing*, 4(2):51–59, 2005. ISSN 1536-1268. doi: 10.1109/MPRV.2005.29.

Sherif Rashad, Mehmed Kantardzic, and Anup Kumar. PAC-WHN: Predictive Admission Control for Wireless Heterogeneous Networks. In *2007 IEEE Symposium on Computers and Communications*, pages 139–144. Ieee, July 2007a. ISBN 978-1-4244-1520-5. doi: 10.1109/ISCC.2007.4381633. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4381633`.

Sherif Rashad, Mehmed Kantardzic, and Anup Kumar. MSP-CACRR: Multidimensional Sequential Patterns Based Call Admission Control and Resource Reservation for Next-Generation Wireless Cellular Networks. *2007 IEEE Symposium on Computational Intelligence and Data Mining*, (Cidm):552–559, 2007b. doi: 10.1109/CIDM.2007.368924. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4221348`.

Simona E. Rombo. Optimal extraction of motif patterns in 2D. *Information Processing Letters*, 109(17):1015–1020, August 2009. ISSN 0020-0190. URL `http://www.sciencedirect.com/science/article/pii/S0020019009001926`.

D Sankoff. Matching sequences under deletion-insertion constraints. *Proceedings of the National Academy of Sciences of the United States of America*, 69(1):4–6, January 1972. ISSN 0027-8424. URL `http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=427531&tool=pmcentrez&rendertype=abstract`.

Albrecht Schmidt, Michael Beigl, and Hans-W. Gellersen. There is more to context than location. *Computers & Graphics*, 23(6):893–901, December 1999. doi: 10.1016/S0097-8493(99)00120-X.

Stephan Sigg, Sandra Haseloff, and Klaus David. An Alignment Approach for Context Prediction Tasks in UbiComp Environments. *IEEE Pervasive Computing*, 9(4):90–97, 2010. ISSN 1536-1268. doi: 10.1109/MPRV.2010.23. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5406495`.

Temple F. Smith and Michael S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981. ISSN 0022-2836.

Libo Song, David Kotz, Ravi Jain, and Xiaoning He. Evaluating location predictors with extensive Wi-Fi mobility data. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, volume 00, pages 1414–1424 vol. 2, Hong Kong, China, 2004. ISBN 0780383567.

P. Songram, V. Boonjing, and S. Intakosum. Closed Multidimensional Sequential Pattern Mining. In *Third International Conference on Information Technology: New Generations (ITNG'06)*, pages 512–517, Las Vegas, NV, USA, 2006. Ieee. ISBN 0-7695-2497-4. doi: 10.1109/ITNG.2006.41. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1611644`.

Ramakrishnan Srikant and Rakesh Agrawal. Mining generalized association rules. In *Proceeding of the 21st international conference on very large data bases (VLDB'95)*, pages 407–419, Zurich, Switzerland, 1995.

Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *EDBT'96 Proceeding of the 5th international conference on extending database technology: Advances in Database Technology*, pages 3–17, Avignon, France, 1996. Springer-Verlag London, UK.

Mark P. Styczynski, Kyle L. Jensen, Isidore Rigoutsos, and Gregory Stephanopoulos. BLOSUM62 miscalculations improve search performance. *Nat Biotech*, 26(3):274–275, March 2008. ISSN 1087-0156. doi: 10.1038/nbt0308-274.

Christian Voigtmann, Klaus David, Hendrik Skistims, and Alexander Roßnagel. Legal assessment of context prediction techniques. *2012 IEEE Vehicular Technology Conference (VTC Fall)*, pages 1–5, September 2012. doi: 10.1109/VTCFall.2012.6399381. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6399381`.

Jianyong Wang and Jiawei Han. BIDE: efficient mining of frequent closed sequences. In *Proceedings of the 20th International Conference on Data Engineering, 2004.*, pages 79–90, 2004. ISBN 1063-6382 VO -. doi: 10.1109/ICDE.2004.1319986.

Jingjing Wang and Bhaskar Prabhala. Periodicity Based Next Place Prediction. In *Nokia Mobile Data Challenge - Next Place Prediction*, 2012.

Wei Wang, Jiong Yang, and Philip S. Yu. Mining patterns in long sequential data with noise. *ACM SIGKDD Explorations Newsletter*, 2(2):28–33, December 2000. ISSN 19310145. doi: 10.1145/380995.381008. URL `http://portal.acm.org/citation.cfm?doid=380995.381008`.

Janet L. Wesson, Akash Singh, and Bradley van Tonder. Can Adaptive Interfaces Improve the Usability of Mobile Applications? *Human-Computer Interaction*, pages 187–198, 2010.

Xifeng Yan, Jiawei Han, and Ramin Afshar. CloSpan: Mining closed sequential patterns in large datasets. In *Proc. 2003 SIAM Int'l Conf. Data Mining (SDM'03)*, pages 166–177, 2003.

Guizhen Yang. The complexity of mining maximal frequent itemsets and maximal frequent patterns. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '04, pages 344–353, New York, NY, USA, 2004. ACM. ISBN 1-58113-888-1. doi: 10.1145/1014052.1014091. URL `http://doi.acm.org/10.1145/1014052.1014091`.

Jiong Yang, Wei Wang, Philip S. Yu, and Jiawei Han. Mining long sequential patterns in a noisy environment. *Proceedings of the 2002 ACM SIGMOD international conference on Management of data - SIGMOD '02*, 4(d):406, 2002. doi: 10.1145/564736.564738. URL `http://portal.acm.org/citation.cfm?doid=564691.564738`.

Zhenglu Yang and Masaru Kitsuregawa. LAPIN-SPAM: An Improved Algorithm for Mining Sequential Pattern. In *21st International Conference on Data Engineering Workshops (ICDEW'05)*, pages 1222–1222. Ieee, 2005. ISBN 0-7695-2657-8.

doi: 10.1109/ICDE.2005.235. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1647839.

Zhenglu Yang, Yitong Wang, and M Kitsuregawa. LAPIN: effective sequential pattern mining algorithms by last position induction for dense databases. In R. Kotagirim, P. R. Krishna, M. Mohania, and E. Nantajeewarawat, editors, *12th International Conference on Database Systems for Advanced Applications, DASFAA 2007*, volume 1, pages 1020–1023, Bangkok, Thailand, 2007. Springer Berlin Heidelberg. URL http://link.springer.com/chapter/10.1007/978-3-540-71703-4_95.

Mariko Yoshida, Tetsuya Iizuka, Hisako Shiohara, and Masanori Ishiguro. Mining sequential patterns including time intervals. volume 4057, pages 213–220, 2000. URL http://dx.doi.org/10.1117/12.381735.

Mohammed J. Zaki. Efficient enumeration of frequent sequences. In *Proceedings of the seventh international conference on Information and knowledge management*, pages 68–75. ACM, 1998. URL http://dl.acm.org/citation.cfm?id=288643.

Mohammed J. Zaki. Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, 12(3):372–390, 2000. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=846291.

Mohammed J. Zaki. SPADE: An Efficient Algorithm for Mining Frequent Sequences. *Machine Learning*, 42(1-2):31–60, 2001. ISSN 0885-6125. doi: 10.1023/A:1007652502315. URL http://dx.doi.org/10.1023/A:1007652502315.

Qiankun Zhao and Sourav S. Bhowmick. Sequential pattern mining: A survey. Technical Report 2003118, CAIS Nayang Technological University Singapore, 2003. URL http://www.textedu.ru/tw_files2/urls_6/147/d-146938/7z-docs/5.pdf.

Zhou Zhao, Da Yan, and Wilfred Ng. Mining Probabilistically Frequent Sequential Patterns in Large Uncertain Databases. *IEEE Transactions on Knowledge and Data Engineering*, 99(PrePrints):1, July 2013. ISSN 1041-4347. doi: 10.1109/TKDE.2013.124. URL http://doi.ieeecomputersociety.org/10.1109/TKDE.2013.124.

Yu Zheng, Lizhu Zhang, Zhengxin Ma, Xing Xie, and Wei-Ying Ma. Recommending friends and locations based on individual location history. *ACM Transactions on the Web*, 5(1):1–44, February 2011. ISSN 15591131. doi: 10.1145/1921591.1921596. URL http://portal.acm.org/citation.cfm?doid=1921591.1921596.

Rui Feng Zhu and Tadao Takaoka. A technique for two-dimensional pattern matching. *Communications of the ACM*, 32(9):1110–1120, 1989. URL `http://dl.acm.org/citation.cfm?id=66459`.

Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *Information Theory, IEEE Transactions on*, 24(5):530–536, 1978. ISSN 0018-9448.

Benchmarks of Supervised Classification Algorithms for Next Place Prediction

A.1 The Next Place Prediction Problem

This annex illustrates our efforts undertaken in the context of the Nokia Mobile Data Challenge (MDC, Laurila et al. [2012]) task 2 "Next Place Prediction". Our aim is to assess the performance of some well explored statistical algorithms chosen according to our expectations of their suitability, as well as classic "benchmark" approaches. This allows the selection of the most accurate algorithm for the prediction task, and a general judgement on the suitability of the approaches for this task.

A step of pre-treatment of the MDC dataset is required to generate a subset of data that is suitable for training a statistical model using the algorithms explored. This treatment consists of a combination of two global – being applied equally to all users – filters, the first of which is an *a priori* "2D" feature selection, and the second a projection of these values into a 1D feature vector. Subsequently two statistical methods of feature selection are evaluated on this vector on per user basis. This pre-processing step is documented in the second section. The third section contains a brief presentation of the algorithms examined and their respective evaluation results. The final section summarizes the results and contains our conclusions.

A.2 Dataset Analysis

The MDC dataset [Laurila et al., 2012] is highly heterogeneous in nature. This stems both from the peculiarities of the group providing the data, and from the challenges of leading a large scale acquisition campaign. The most obvious heterogeneity lies in the

difference of sizes of the per-user data sets: around 620 MB±320 MB in human readable, tabulator separated value format. Additionally, different users have made different use of the phone capabilities, leading to different distributions of useful data, e.g. some users not using the calendar functionality of their phone, and others using it extensively. The size of the dataset is also an important quantity: the "width" – i.e. the number of different data sources – is large (around 75 features), but the actual number of training cases (per individual) for the next place prediction task is fairly small (between 100 and 1500). This means that statistical analysis is more likely to be influenced by outliers, potentially leading to inadvertent overfitting of models and in general a higher error for the smaller sub-datasets. The data itself can also be unreliable: there are intervals of missing data and occasionally the data does not pass basic sanity tests, e.g. a time zone changing by many hours, *during* a single visit. The hardware homogenity at least allows for some transversal coherency of the gathered data, but this is of little consequence for the next place prediction.

We propose a three-step approach for the preparation of this data, that consists of two *a priori* selection, and a final statistic filtering, using either the consistency criterion proposed by Liu and Setiono [1996] or a correlation-based feature selector (Cfs, [Hall, 1999]). In the first step we select twelve features, that should *influence* or *indicate* the decision which location is going to be visited next, according to our global conceptual model. From the 75 features available, this is the subset used as base of the model:

- hour of the day and day of the week – based on the assumption that certain visits will have a regularity in time, which is the case in non-shift workers and students;
- place id – based on the assumption, that some places are visited exclusively after a visit of another place;
- bluetooth devices – can link to sub-locations of a visit, transportation choice or the social environment;
- applications used – a mapping application may strongly correlate with first-time visits, the messaging application may provide information on a social link;
- call / message type – an outgoing or incoming phone call or message influence the choice of next location;
- call duration – the length of a phone call is expected to be an indicator of the social link between call participants;
- call / message contact – the contact can be an indicator of the next destination;

- movement of the phone – is an indicator of physical activity, which could indicate a destination;
- charging state – if the phone is charged at a place, it can be expected that the next place is less likely to have charging facilities available, or that a long transition will follow;
- battery level – if the battery level at departure is low, the next location is likely to be a place with charging facilities, and close by;
- calendar event titles – both the knowledge that a calendar event is linked to the current place and visit, as well as the next known calendar event are clearly hints at the following destination.

As most of these values are time dependant during a visit, the next step is to project the key information of these twelve features into a single characteristic vector. Here again the choices were made based on expected utility and with the goal of minimizing the features to limit noise and computational effort required.

Per visit, the hour of day and day of week values were retained for both the beginning and end of the visit. The two most frequently encountered bluetooth addresses were recorded, as well as the two most frequently used phone applications. The details of the last communication (incoming/outgoing, message/call and contact ID), the sum of all acceleration values with a log weighting favouring the end of the visit, the portion of the visit that the phone was connected to the charger, the mean battery level as well as the battery level at the end of the visit and any calendar event during the following 24 hours and the first calendar event planned for the current visit.

On average 59.6% ($\pm$14.3) of recorded visits were to the two most visited locations. More than one third of visits, 37.2% ($\pm$11.2), were to the most visited location. This value represents a lower boundary for prediction precision, when using the training set for evaluation. Around one sixth of the visits were to places that had an incidence rate below one percent. These were grouped as a single location, which, when predicted, are considered to be a new location.

A.3 Next Visit Prediction

It is currently beyond the scale of physical and logical modelling to emulate the decision-making process of a person, even if their complete context and history are known. In addition our knowledge of users and their context are imperfect. Even logical links

Table A.1.: Classifier and Filter Configurations Tested

Classifier	Classifier Settings	Explicit Multiclass Schemes
NaiveBayes [John and Langley, 1995] ("NB")	Kernel estimator	none 1-1 ("M3") 1-all ("M0")
LADTree [Holmes et al., 2001] ("LAD")	10 or 20 Boosts	none 1-1 ("M3") 1-all ("M0")
SMO [Platt, 1999]	Complexity parameter C = 0.5 or 10 Polynomial Kernel with exponent E = 1.0 or 2.0	none 1-1 ("M3") 1-all ("M0")

between a decision and known context cannot be certainly established; at best different correlation measures could be calculated. As a result, there is no clear *a priori* indication which statistical classification model is best applicable to this process. Hence an empirical, iterative approach is the way we choose to obtain a predictor with optimal accuracy. A common evaluation scenario similar to the test scenario was chosen, dividing the training set along a 90%/10% split. The first 90% were used as training set, which was then evaluated on the other 10%. The evaluation metric is the ratio and number of correct predictions.

Our interpretation of the problem as a multi-class classification problem, with asymmetrically sized classes and a mixed characteristic vector, containing both numerical and nominal values, permitted us to evaluate the naive Bayesian approach [John and Langley, 1995], the LogitBoost alternating decision (LAD, [Holmes et al., 2001]) tree classifier and a support vector machine based classifier, the sequential minimal optimization (SMO, [Platt, 1999]) algorithm. These methods represent three different approaches to statistical modelling and are all flexible enough to accept our dataset without modification.

The specific configurations are referenced in Table 1. Additionally we tested four configurations of the feature vector, a basic vector using just the time and location information of the previous visit (time of day and day of week of both beginning and end of the visit and location), the extended vector (labeled "full" in the graphs) including the context data mentioned above, and two reduced vectors, using the features selected by either the correlation based feature selection filter (labelled "Cfs") or consistency based feature selection [Liu and Setiono, 1996] filter (labelled "Consist"). For each filter

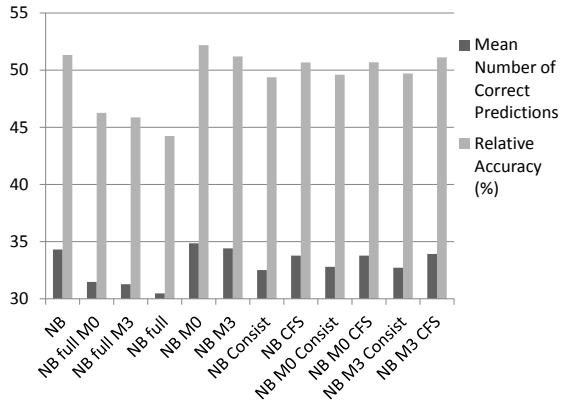

Figure A.3.1.: Results for the Naive Bayes Classification Algorithm

we were able to use an exhaustive search algorithm, due to our limited feature set size. These four variants allow us to show the impact of the availability of additional context information as well as what can be achieved by filtering on a per user level.

To cater to the multi-class nature of the task, we used each algorithm's natural multi-class approach, and also put in place two variants that forced the classifiers to function in 1-against-all (labelled M0 in the graphs) and 1-against-1 (labelled M3) mode. For our experiments, we used the Weka statistics suite [Hall et al., 2009].

In the following we present the results of our evaluation of a total of 84 different configurations. We constrain ourself to two evaluation scores: mean absolute accuracy, i.e. the average number of correct predictions, and mean relative accuracy, i.e. the average percentage of correct predictions. The former is a closer indicator of the algorithm's performance for the challenge, the latter is a better measure of overall user experience. Significantly larger numbers of boost iterations for the LADtree could not be used, as the data set for some users is too small. The partial results we did obtain were not indicative of precision enhancements. Similarly, our attempts to run exhaustive error correction multiclass classifiers were foiled by the large size of other user's data sets, which required more than 4GB of memory.

Looking at the results of the naive Bayes classifier (see Fig. A.3.1), it becomes obvious, that it is negatively impacted (losing around 5 percent points of average accuracy) by the additional context information, unless it is filtered. Otherwise, results vary very little: mean accuracy is between 49% and 52%, the average number of correct predictions varies between 30 and 31 for the full feature vector and between 32.5 and 35 in the other configurations. Specifically, the minimalist dataset has the same average accuracy as the feature selection filtered variants of the full dataset.

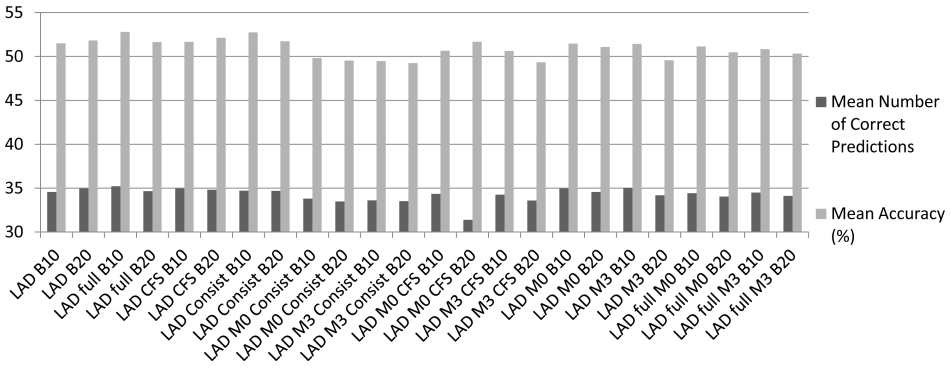

Figure A.3.2.: Results for the LogitBoost Adaptive Decision Tree Classification Algorithm

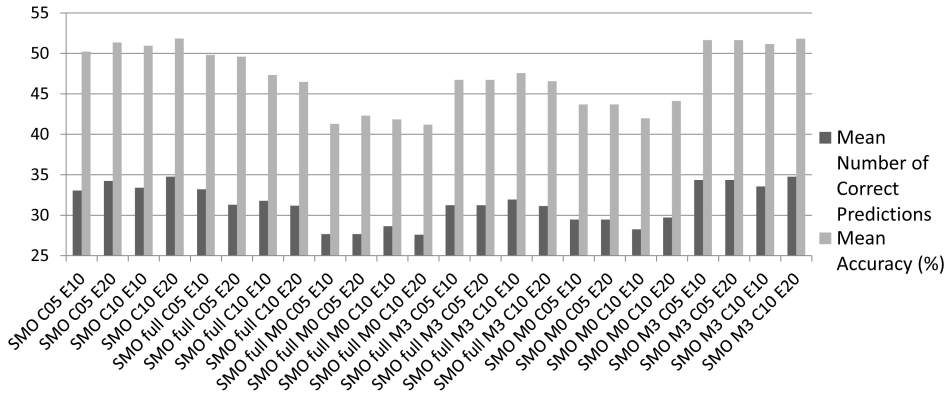

Figure A.3.3.: Results for the SMO Classification Algorithm without Feature Selection Filters

The LADtree (see Fig. A.3.2) model is the most stable of all tested algorithms, and appears to perform almost independently of the additional data. Mean accuracy is consistently between 49 and 52 percent and there are between 33 and 35 (one outlier at 31.4) average correct predictions. Two configurations (10x Boosting on the full dataset and 10x Boosting on the minimal dataset in a 1 - 1 multiclass classifier) exceed a mean correct prediction count of 35, equivalent to over 2800 correct predictions on our test dataset. Overall, LADtree performs the most consistent manner and is the most accurate, but only barely exceeds the results from the Naive Bayesian approach.

Most of the results of our evaluations of the SMO algorithm (see Figs. A.3.3 and A.3.4) are disappointing, especially when using any of the two tested feature selection filters. Also, using a 1 - all multi-class approach decreased accuracy noticeably, leading to some

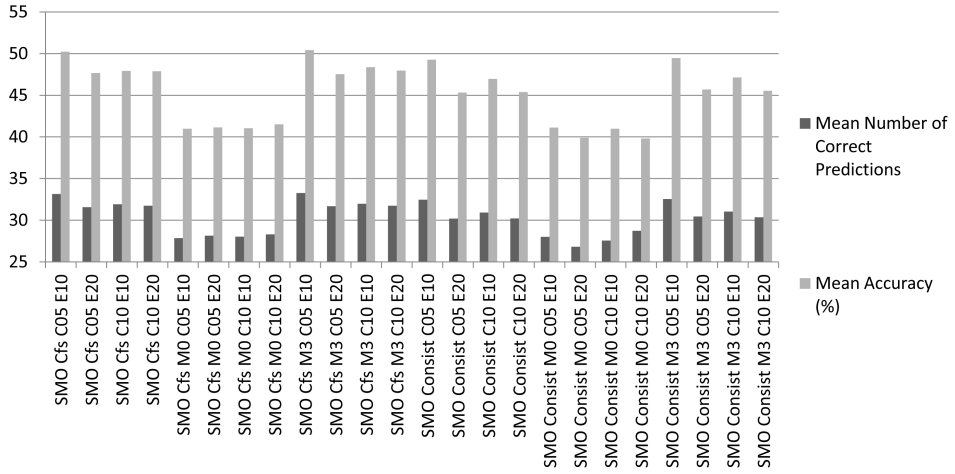

Figure A.3.4.: Results for the SMO Classification Algorithm with Feature Selection Filters

of the weakest results of our evaluation. Nonetheless, without feature selection filters, on the reduced dataset, and either forced 1 - 1 multi-class classification or the native 1 - 1 multi-class approach of SMO, results exceeding 50% mean accuracy could be achieved. Any additional data in the training set decreased accuracy.

A.4 Conclusion

The approaches we examined here only show marginal differences in performance, when only the most accurate configurations are taken into account. Nonetheless, the overall best algorithm in our testing set of three is the LADtree classifier. Regarding the feature selection, correlation based feature selection appears to perform consistently better than consistency based filtering. Also, a LogitBoost decision tree model performs slightly better with 10 boosting iterations, than with 20 iterations - in the majority of cases. The around 50% accuracy of the predictions observed during our evaluations, lead us to believe that a dedicated hybrid modelling approach, optimised just to predict the two most frequently visited locations, may achieve higher accuracies than we obtained during our series of evaluations. Given the constrained time resources given by the challenge, and the complexity and heterogeneity of the dataset, we decided to first examine the necessity of following a dedicated modelling approach. One example of a possible adaptation, would be the development of dedicated estimators for each feature, when using a naive Bayes classifier. For example the cyclic nature of the temporal

features are currently not being correctly taken into account.

Another important observation is the fact that adding what appeared to be – from an *a priori* stand point – a useful set of features extracted from the context to the classification process has little to no value in all three prediction models we examined. The relatively low number of training instances makes observation of strong correlations unlikely, which in turn reduces the predictive capability of any model based on this data. While our examination was by no means exhaustive, we see no basis to expect major gains using a similar approach, with more extensive, iterative tuning. It therefore appears that the extended conceptual model we based our feature extraction process on, was not matched by any of the statistical models we evaluated during this work.

Details on the evaluation processes

B.1 Real world data

B.1.1 Similarity score tables

These are the similarity scores used for the alignment of our real world dataset. The value 0 in each table represents the place holder value, if a sensor had no (valid) reading at this point in time.

Table B.1.: Similarity score table for luminance

	0	1	2	3	4	5	6	7	8	9	10
0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	4	-9	-8	-9	-9	-10	-9	-10	-10	-10
2	-1	-9	3	-7	-8	-9	-9	-9	-9	-9	-9
3	-1	-8	-7	3	-8	-9	-10	-9	-9	-9	-10
4	-1	-9	-8	-8	3	-9	-10	-9	-9	-10	-10
5	-1	-9	-9	-9	-9	3	-9	-9	-9	-9	-8
6	-1	-10	-9	-10	-10	-9	1	-10	-7	-10	-7
7	-1	-9	-9	-9	-9	-9	-10	1	-7	-10	-8
8	-1	-10	-9	-9	-9	-9	-7	-7	3	-10	-10
9	-1	-10	-9	-9	-10	-9	-10	-10	-10	2	-7
10	-1	-10	-9	-10	-10	-8	-7	-8	-10	-7	2

Table B.2.: Similarity score table for proximity

	0	1	2
0	1	-1	-1
1	-1	2	-5
2	-1	-5	2

Table B.3.: Similarity score table for places

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	6	-9	-10	-9	-9	-10	-10	-9	-10	-10	-10	-9	-9	-10
2	-1	-9	3	-9	-8	-9	-10	-9	-9	-10	-9	-10	-8	-9	-10
3	-1	-10	-9	2	-10	-9	-10	-9	-7	-10	-10	-10	-8	-9	-10
4	-1	-9	-8	-10	3	-8	-9	-9	-8	-10	-10	-10	-8	-9	-10
5	-1	-9	-9	-9	-8	5	-9	-8	-10	-10	-9	-10	-9	-8	-10
6	-1	-10	-10	-10	-9	-9	5	-8	-9	-10	-9	-10	-9	-9	-10
7	-1	-10	-9	-9	-9	-8	-8	5	-9	-10	-10	-10	-10	-9	-10
8	-1	-9	-9	-7	-8	-10	-9	-9	6	-9	-9	-10	-9	-9	-9
9	-1	-10	-10	-10	-10	-10	-10	-10	-9	5	-8	-10	-10	-8	-10
10	-1	-10	-9	-10	-10	-9	-9	-10	-9	-8	6	-10	-10	-8	-10
11	-1	-10	-10	-10	-10	-10	-10	-10	-10	-10	-10	7	-10	-9	-10
12	-1	-9	-9	-8	-8	-9	-9	-10	-9	-10	-10	-10	2	-8	-10
13	-1	-9	-9	-9	-9	-8	-9	-9	-9	-8	-8	-9	-8	5	-9
14	-1	-10	-10	-10	-10	-10	-10	-10	-9	-10	-10	-10	-10	-9	7

Table B.4.: Similarity score table for battery levels

	0	1	2	3	4	5	6	7	8
0	1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	1	-1	-4	-3	-5	-1	-6	-2
2	-1	-1	1	-5	-4	-6	-2	-7	-3
3	-1	-4	-5	1	-1	-1	-3	-2	-2
4	-1	-3	-4	-1	1	-3	-2	-4	-1
5	-1	-5	-6	-1	-3	1	-4	-1	-2
6	-1	-1	-2	-3	-2	-4	1	-5	-1
7	-1	-6	-7	-2	-4	-1	-5	1	-3
8	-1	-2	-3	-2	-1	-2	-1	-3	1

Table B.5.: Similarity score table for orientation values

	0	1	2	3	4	5	6	7	8
0	1	-1	-1	-1	-1	-1	-1	-1	-1
1	-1	2	-8	-7	-8	-4	-7	-7	-9
2	-1	-8	4	-7	-9	-7	-9	-8	-9
3	-1	-7	-7	2	-8	-4	-7	-7	-9
4	-1	-8	-9	-8	4	-8	-9	-9	-8
5	-1	-4	-7	-4	-8	4	-9	-8	-9
6	-1	-7	-9	-7	-9	-9	3	-5	-9
7	-1	-7	-8	-7	-9	-8	-5	4	-9
8	-1	-9	-9	-9	-8	-9	-9	-9	4

B.1.2 Detailed results of the alignment algorithm evaluation on synthetic data

These are the detailed result for the 5 datasets and 27 configurations used for the synthetic evaluation, for the three metrics which have thus far only been presented in aggregated form.

	d-set 1						d-set 2					
	precision		size ratio		recall		precision		size ratio		recall	
index	mean	err	mean	err	mean	err	mean	err	mean	err	mean	err
0	0.42	0.32	4.13	2.73	1.00	0.00	0.91	0.20	1.23	0.65	1.00	0.00
1	0.89	0.21	1.32	0.93	1.00	0.00	0.96	0.10	1.06	0.25	1.00	0.00
2	0.82	0.30	1.82	1.99	1.00	0.00	0.95	0.13	1.09	0.34	1.00	0.00
3	0.89	0.21	1.29	0.81	1.00	0.00	0.97	0.09	1.06	0.24	1.00	0.00
4	0.45	0.39	5.86	5.50	1.00	0.00	0.55	0.33	3.03	2.61	1.00	0.00
5	0.50	0.39	5.25	5.31	1.00	0.04	0.88	0.23	1.38	1.02	1.00	0.00
6	0.68	0.36	2.81	3.25	1.00	0.00	0.92	0.18	1.18	0.55	1.00	0.00
7	0.23	0.26	8.92	5.61	1.00	0.00	0.42	0.33	4.43	3.24	0.98	0.07
8	0.23	0.26	8.99	5.64	1.00	0.00	0.63	0.37	2.97	2.75	0.99	0.06
9	0.54	0.39	4.69	5.07	0.99	0.07	0.79	0.30	1.74	1.46	1.00	0.02
10	0.85	0.27	1.53	1.27	1.00	0.04	0.94	0.15	1.13	0.42	1.00	0.00
11	0.91	0.19	1.23	0.73	1.00	0.00	0.95	0.12	1.08	0.29	1.00	0.00
12	0.51	0.36	3.70	2.86	1.00	0.00	0.70	0.32	1.95	1.24	1.00	0.00
13	0.58	0.37	3.09	2.53	1.00	0.06	0.90	0.20	1.24	0.64	1.00	0.00
14	0.77	0.32	1.89	1.60	1.00	0.00	0.93	0.16	1.16	0.47	1.00	0.00
15	0.39	0.31	4.35	2.68	1.00	0.00	0.58	0.34	2.50	1.51	0.99	0.04
16	0.49	0.36	3.72	2.75	1.00	0.07	0.71	0.32	1.92	1.22	1.00	0.00
17	0.57	0.36	2.98	2.22	1.00	0.02	0.85	0.25	1.41	0.78	1.00	0.01
18	0.39	0.37	8.68	10.54	1.00	0.00	0.89	0.22	1.30	0.77	1.00	0.00
19	0.79	0.32	2.22	3.11	1.00	0.01	0.94	0.14	1.12	0.40	1.00	0.00
20	0.36	0.39	11.26	12.35	1.00	0.00	0.49	0.35	5.38	6.90	1.00	0.00
21	0.48	0.40	8.45	11.74	1.00	0.00	0.86	0.25	1.43	1.14	1.00	0.00
22	0.70	0.38	4.50	8.19	1.00	0.00	0.93	0.17	1.16	0.52	1.00	0.00
23	0.26	0.34	17.55	14.69	1.00	0.00	0.29	0.34	10.44	8.49	0.98	0.08
24	0.26	0.34	17.89	14.89	1.00	0.00	0.30	0.34	10.29	8.45	0.98	0.07
25	0.47	0.40	8.88	12.30	0.99	0.08	0.76	0.32	2.17	2.84	1.00	0.02
26	0.37	0.33	5.83	4.86	1.00	0.00	0.91	0.20	1.24	0.67	1.00	0.00
	d-set 3						d-set 4					
0	0.17	0.10	6.96	2.66	0.96	0.13	0.45	0.32	3.66	2.56	1.00	0.00
1	0.04	0.06	6.13	7.33	0.21	0.26	0.90	0.22	1.33	1.02	1.00	0.00
2	0.15	0.16	6.52	5.70	0.63	0.37	0.77	0.32	2.11	2.39	1.00	0.02
3	0.08	0.11	5.28	4.58	0.34	0.35	0.91	0.21	1.27	0.81	1.00	0.00
4	0.11	0.10	6.93	4.86	0.68	0.42	0.30	0.33	8.03	5.72	1.00	0.00
5	0.16	0.20	4.17	4.20	0.48	0.44	0.49	0.38	5.33	5.42	0.99	0.10
6	0.13	0.15	4.40	3.29	0.52	0.44	0.70	0.36	2.77	3.16	1.00	0.02
7	0.10	0.05	10.07	6.50	0.78	0.31	0.30	0.31	7.36	5.34	1.00	0.00
8	0.10	0.06	10.34	6.51	0.81	0.28	0.31	0.32	7.34	5.37	1.00	0.00

9	0.20	0.31	2.80	1.56	0.41	0.47	0.50	0.38	4.97	5.14	0.99	0.09
10	0.09	0.09	4.22	2.58	0.42	0.42	0.82	0.28	1.58	1.22	1.00	0.00
11	0.06	0.12	3.59	2.24	0.25	0.39	0.91	0.21	1.24	0.69	1.00	0.00
12	0.15	0.10	4.33	2.22	0.71	0.43	0.43	0.32	3.98	2.72	1.00	0.00
13	0.23	0.18	3.41	2.15	0.69	0.40	0.60	0.37	2.93	2.35	0.99	0.12
14	0.16	0.20	3.27	2.01	0.46	0.45	0.75	0.32	1.92	1.59	1.00	0.00
15	0.13	0.06	5.94	2.48	0.78	0.33	0.49	0.34	3.60	2.60	1.00	0.00
16	0.23	0.18	2.95	1.49	0.63	0.40	0.50	0.36	3.71	2.75	0.98	0.13
17	0.26	0.28	2.97	1.71	0.60	0.42	0.62	0.37	2.87	2.39	1.00	0.00
18	0.05	0.05	12.08	7.45	0.51	0.18	0.16	0.20	17.17	14.66	1.00	0.00
19	0.06	0.08	7.89	8.96	0.27	0.26	0.73	0.35	2.92	4.75	1.00	0.03
20	0.09	0.09	11.88	11.49	0.64	0.39	0.20	0.29	18.45	14.15	1.00	0.00
21	0.12	0.15	5.31	5.28	0.44	0.41	0.44	0.40	9.92	12.32	1.00	0.04
22	0.10	0.13	5.72	5.24	0.41	0.41	0.67	0.39	5.05	8.81	1.00	0.04
23	0.10	0.09	17.29	16.14	0.83	0.23	0.15	0.24	21.38	14.20	1.00	0.00
24	0.14	0.18	16.43	16.20	0.80	0.26	0.15	0.24	21.37	14.21	1.00	0.00
25	0.17	0.24	3.09	1.96	0.44	0.44	0.44	0.39	9.77	12.88	0.99	0.08
26	0.08	0.07	7.55	3.85	0.56	0.25	0.27	0.28	7.52	5.27	1.00	0.00

		d-set	5			
0	0.82	0.24	0.33	0.14	0.25	0.06
1	0.48	0.11	0.39	0.12	0.18	0.04
2	0.67	0.34	0.36	0.13	0.24	0.13
3	0.42	0.12	0.41	0.13	0.16	0.04
4	0.46	0.11	0.41	0.14	0.18	0.04
5	0.52	0.10	0.39	0.11	0.19	0.04
6	0.37	0.10	0.48	0.09	0.17	0.03
7						
8						
9	0.29	0.03	0.49	0.09	0.14	0.02
10	0.77	0.26	0.38	0.16	0.28	0.10
11	0.65	0.36	0.36	0.12	0.24	0.14
12	0.44	0.10	0.38	0.12	0.16	0.04
13	0.46	0.13	0.46	0.15	0.20	0.04
14	0.44	0.10	0.34	0.10	0.15	0.04
15	0.42	0.14	0.48	0.14	0.18	0.01
16	0.81	0.24	0.34	0.17	0.25	0.06
17	0.75	0.26	0.40	0.18	0.28	0.10
18	0.66	0.35	0.36	0.14	0.24	0.13
19	0.43	0.11	0.40	0.13	0.16	0.04
20	0.51	0.11	0.35	0.10	0.17	0.03

21	0.33	0.09	0.49	0.13	0.16	0.03
22	0.31	0.11	0.55	0.18	0.15	0.03
23	0.39	0.04	0.36	0.05	0.14	0.03
24	0.82	0.24	0.34	0.16	0.25	0.06
25	0.75	0.27	0.39	0.17	0.28	0.10
26						

B.1.3 Expert annotated patterns

These are the reference patterns identified through visual examination of the dataset of one participant of our data collection campaign. They were selected through side by side comparison of data for all days, and identifying similar visual structures in the bar diagrammes. Pattern 1 (Fig. B.1.1) corresponds to the campaign participant spending a weekend evening and night at the out-of-town second home. Pattern 2 (Fig. B.1.2) is one of the instances of the campaign participant getting up in the morning and going to work. Pattern 3 (Fig. B.1.3) is slightly more tenuous: it covers only the proximity sensor and location, and corresponds to the campaign participant being at work and busy, hence the phone is stored away, with the screen covered.

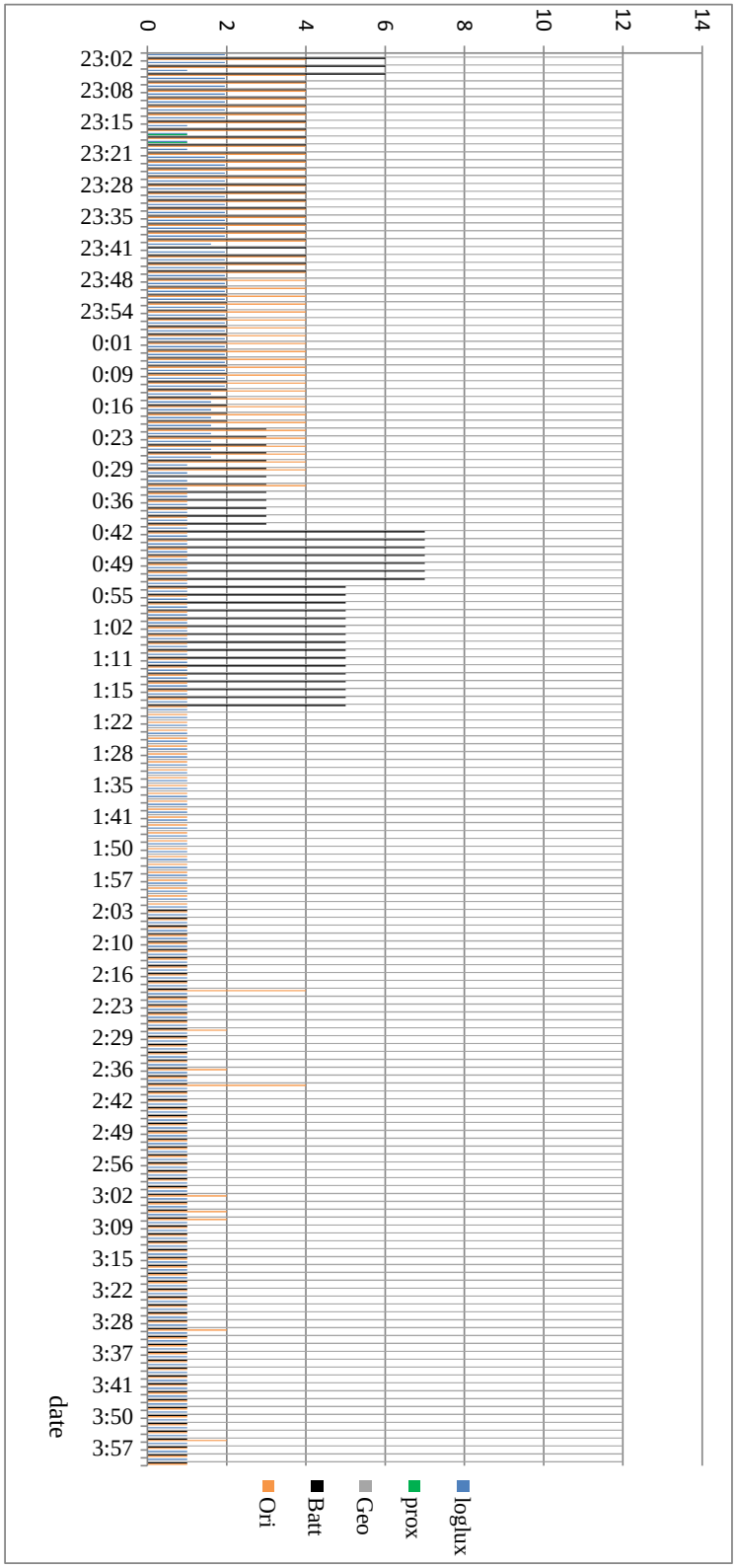

Figure B.1.1.: Reference pattern 1 - evening and night at the week-end home.

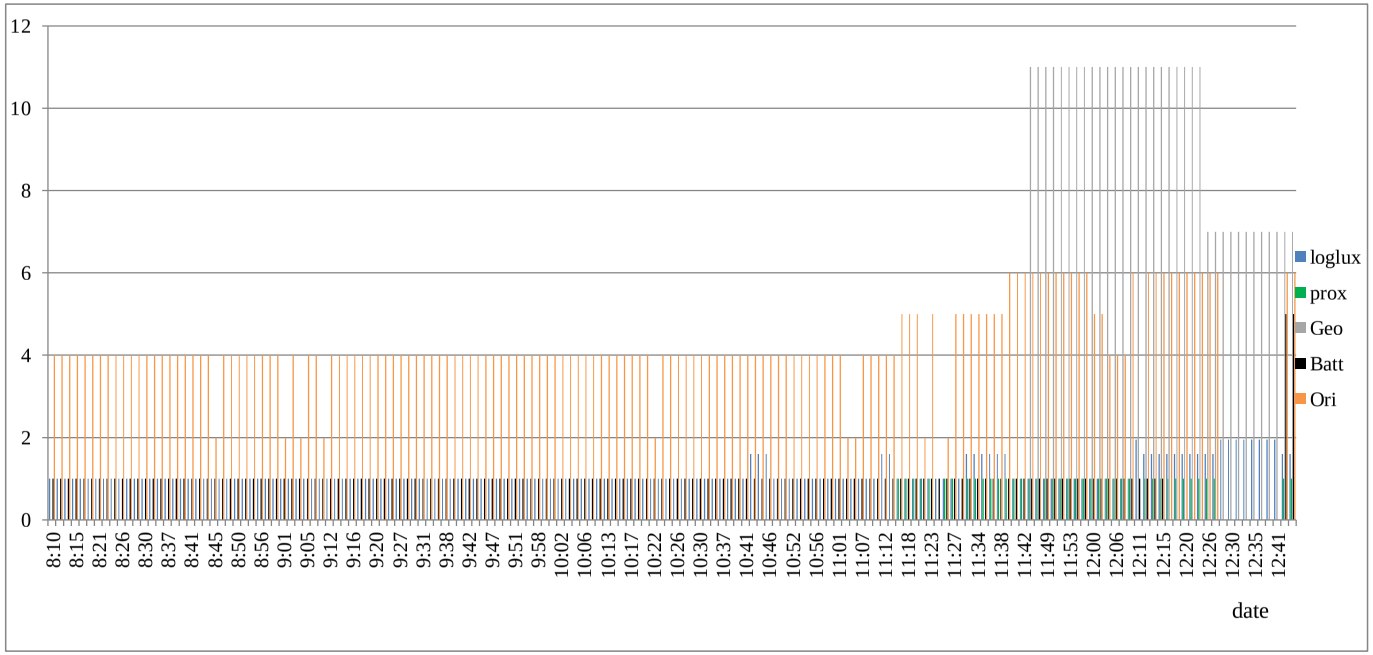

Figure B.1.2.: Reference pattern 2 - morning and transit to work on a week-day.

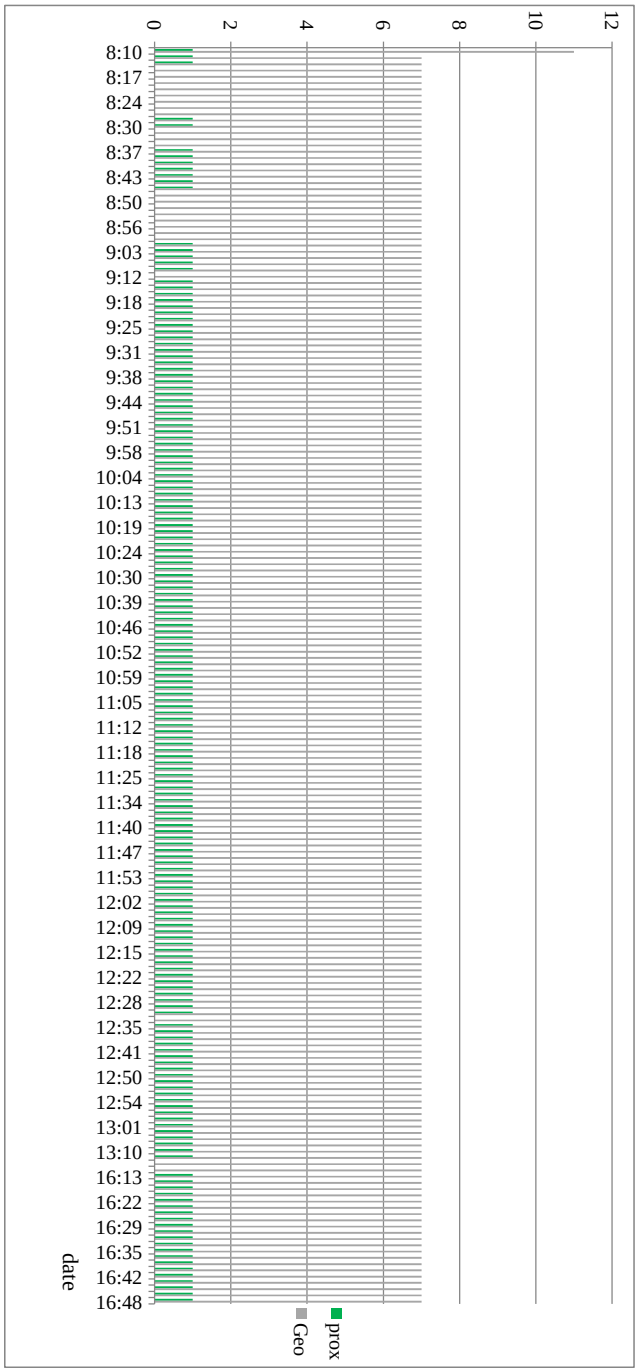

Figure B.1.3.: Reference pattern 3 - at work, telephone stored or turned over.