Metrics and visualisation for crime analysis and genomics

Laros, J.F.J.

Chapter 7

Alignment of Multiset Sequences

The concept of multiset sequences is common in a number of different application domains. This chapter introduces a new metric for the similarity between these sequences. Various types of alignments are used to find the shortest distance between two sequences. This distance is based on a well-defined distance measure for multisets.

Employing this, a pairwise distance can be defined for two sequences. Apart from the pairwise distances, the occurrence of holes (for timestamped sequences) can also be used in determining similarity; several options are explored. Applications of this metric to the analysis of criminal careers and access logs are reviewed.

7.1 Introduction

Data mining techniques are often employed to extract information from large quantities of data [67]. One of the recurring concepts in data mining is that of *multisets* (also known as bags). A multiset is a set over a finite alphabet where elements may be present more than one time. For example, a vase filled with coloured marbles is a multiset, there can be more than one marble of a particular colour in the vase. Another common theme is data with a strict ordering in some sense, i.e., a temporal aspect. Such information is usually in the form of timestamps.

In *bio-informatics* [26, 65], and especially the field of genomics, the concept of *alignment* is well-known. Various types of alignments are used to match specific strings of DNA to each other. They are usually employed to measure how different the target string is from the other strings in the experiment. The more different it is, the more interesting it might be. If two strings of DNA are close to each other in this sense, there is a reasonable chance that these strings will stick to each other (or hybridise as the biologists call it), thus ruining the experiment.

Strings that have a very high edit distance to all other strings (within a specific genome) are therefore a research area with many applications, the construction of *primers* [17, 37] and *micro arrays* [58] being two of them.

A good example dealing with ordered multiset sequences is the analysis of *criminal careers*, a concept used in the law enforcement area [50, 51].

We can represent a criminal career (the criminal activities a person exhibits throughout his or her life) as a sequence of multisets. Each element (of a multiset) representing a crime. Since someone can commit multiple crimes and also repeat a crime within a certain timeframe, the adoption of the multiset paradigm is natural. In the analysis of criminal careers, the temporal aspect of the elements in a career (the crime is committed on a certain date) is also an important characteristic. Combining the properties of this distribution with the frequency of the crimes can provide valuable insight into the behaviour of criminals. By calculating distances between careers, one can make predictions about the threat someone poses at the time of arrest, or an analyst could make predictions about the future development of a starting delinquent.

Another research area concerned with the analysis of time sequences is knowledge discovery from (web) access logs, e.g., the activity at a certain time is related to the parts of the world where it is day or night. Also, crawlers and spiders tend to cause a more or less cyclic pattern.

Our method aims to calculate the distance between two sequences of multisets in order to provide an insight in similarities between, e.g., careers or web browsing behaviour. We explore different kinds of alignment and methods of representation of the multiset sequences in order to obtain different visualisations and clusterings, each one accentuating a different aspect of the sequences under consideration. We use the visualisation just as a means to show the fertility of the alignment method.

Apart from the direct applicability in the above mentioned analysis, this new metric can be used as a stepping stone for the enhancement of existing techniques, a good example being temporal extrapolation [14], that depends upon valid metrics.

The overview of the rest of this chapter is as follows. In Section 7.2 we provide information about the concept of alignment and explain the different types of alignment used within our approach. In Section 7.3 we explain how we adapt current alignment methodologies to work with multisets and how they deal with missing data. In Section 7.4 we describe the datasets on which we can apply our techniques and we mention the details of the application of the techniques on the datasets. We conclude in Section 7.5.

7.2 Background

Research of the alignment of strings has been prominent for a number of years, resulting in the development of a wide variety of algorithms. Most of them are calculating the difference between two sequences, which can be done by calculating the so-called *edit distance* [45]. This is defined by the minimum number of edits needed to transform one string into an other. Most often, an *edit* is defined as either an insertion, a deletion or the rewriting of a symbol. Each of these three operations leads to a certain *penalty* that can be different and may even depend on the symbols in the sequence itself. The sum of these penalties is then an upper bound for the edit distance. If the penalty is minimal (usually this is also a minimum number of edits), we speak of *the* edit distance.

The goal of alignment is to minimize this distance for two sequences, in practice arranging the sequences in such a way that the similarities between the two sequences are "placed" below each other in a visualisation. Such a placement is called a *layout*. In a layout *gaps* (usually denoted by `-`'s) represent insertions or deletions. By definition, (and with slight abuse of language) an *alignment* is one of the optimal layouts possible. The problem of constructing an alignment starts with the calculation of the edit distance, yielding a number of different solutions. One of them, selected arbitrarily, is then traced back to generate an alignment. Depending on the purpose of the algorithm, there are several types of alignment available.

A *global alignment* [54] is a form of alignment that has a high preference for matching entire sequences to each other. This makes a global alignment most useful when aligning two sequences that have roughly the same size. It can also be used when there is a good reason within the application domain to assume that short sequences are different from long sequences. By using global alignment, the distance between such sequences will be large by default.

A *local alignment* [66] puts less penalty on gaps at the beginning or end. Therefore it is most suitable to align small strings to large ones. The small string will be matched to the most compatible substring of the large one.

We will not discuss the inner workings of these (well-known) algorithms here. Both global and local alignments are efficiently calculated by employing *dynamic programming* techniques in quadratic time ($\mathcal{O}(mn)$) and usually in linear space ($\mathcal{O}(\min(m,n))$) [31], where m and n are the lengths of the sequences.

Related work can be found in the alignment of event sequences [49] and musical sequence comparison [36], where raw events, rather than bags of events, are considered as elementary units. An other difference with these techniques is that we operate within a discrete domain, whereas other techniques operate on continuous spatio-temporal data, where raw events can be identified, and the need for the use of multisets is absent.

Example 1. In this example we have two sequences $U =$ `AABABBBAA` and $V =$ `AAAABBAA` of consecutive symbols (or atomic actions) from the alphabet {`A`, `B`}. All penalties equal 1. First an alignment is constructed for the strings U and V; one of the possible alignments is depicted below:

```
     012345678
U =  AABABBBAA
     || ||| ||
V =  AAAABB-AA
```

This alignment (an optimal layout) has two errors, one at position 2 and one at position 6. Other alignments with two errors are also possible. □

In the case of alignment, the edit distance is the distance measure for two sequences of symbols. The first requirement of such a distance measure is stating the difference between the symbols (a distance for the elements, from which the edit distance follows). A distance matrix like the one below is typical:

$$\begin{array}{c|ccc}
 & \texttt{A} & \texttt{B} & \texttt{-} \\
\hline
\texttt{A} & 0 & x & y \\
\texttt{B} & x & 0 & z \\
\texttt{-} & y & z & -
\end{array}$$

Here $x, y, z \in \mathbb{R}_{\geq 0}$ are domain specific constants. So in an alignment where `A` has to be matched to `B`, we get a penalty x, and when `B` is matched to a gap, then the penalty is z. Since the matching of two gaps never occurs, this value is omitted from the distance matrix.

Normally a distance matrix like the one from the example will suffice for an alignment. In genomics there are various distance matrices to give a distance between two nucleotides, based upon different experimental data.

When dealing with crimes, where for example `A` is a bicycle theft and `B` is robbery, a domain expert should specify y by giving the absolute severity of `A` and z by giving the absolute severity of `B`. The maximum or average sentence for each crime might be a reasonable value (in days of captivity for example). The value of x can then be set to the difference in severity between crimes `A` and `B`.

In the case of access logs, we can assign different distances to elements according to their source. For example, if a certain network is known for its many bots, we can increase or decrease the distance to other elements depending on what we want to investigate. We can also do this for countries if we want to accentuate (or under-emphasize) hits from a certain area. This can potentially be realised by using a GIS database.

The main bottleneck in this method for multiset strings is that using a predefined distance matrix is not feasible because the number of multisets is too large or even infinite. Since each symbol may be used arbitrarily often, there are infinitely many multisets. For example, a criminal can steal several (potentially very many) bikes in one year.

In this chapter we present a solution to this problem and suggest some (domain specific) distance measures.

7.3 Alignment adaptation

For the alignment of general structures, we need at least two things: a valid distance measure for the elements (in our case multisets) of the structure and a way to deal with so-called “holes”.We first discuss the pairwise comparison and then the conversion into sequences of multisets.

Pairwise comparison

In order to do alignment on structures that are not bare sequences a distance matrix with pre-calculated values might not be sufficient any more. In our case, we want to know the distance between multisets. To do this, we use the following approach (see Chapter 6) which generalises well-known distance measures like the Jaccard [34] and the Canberra distance.

Let f be a function $f : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ with finite supremum M and the following properties:

$$\begin{array}{lcll} f(x,y) & = & f(y,x) & \text{for all } x,y \in \mathbb{R}_{\geq 0} \\ f(x,x) & = & 0 & \text{for all } x \in \mathbb{R}_{\geq 0} \\ f(x,0) & \geq & M/2 & \text{for all } x \in \mathbb{R}_{>0} \\ f(x,y) & \leq & f(x,z)+f(z,y) & \text{for all } x,y,z \in \mathbb{R}_{\geq 0} \end{array}$$

For a multiset X, let $S(X)$ denote its underlying set. For multisets X, Y over the set $\{1,2,\ldots,n\}$ we define (if $X \neq \varnothing$ or $Y \neq \varnothing$)

$$d_f(X,Y) = \frac{\sum_{i=1}^{n} f(x_i, y_i)}{|S(X) \cup S(Y)|}$$

and $d_f(\varnothing,\varnothing) = 0$. Here x_i, resp. y_i, denote the integer number of elements in category i for multiset X, resp. Y.

As function f we will use the following function ($x, y \in \mathbb{R}_{\geq 0}$, both ≥ 1, or $x = 0$ or $y = 0$):

$$f(x,y) = \frac{|x-y|}{(x+1)(y+1)}$$

(Other values with $x < 1$ or $y < 1$ are not needed.) In Chapter 6 it was shown that this d_f is a decent distance measure; note that the supremum M equals 1. The reason that we use this particular function is because the difference between two different small numbers (like 0 and 1) is large in comparison to the difference between two distinct large numbers. For our purposes, this is natural, because the fact that someone has committed a crime is more important than the number of crimes that were committed in that category. Indeed, the fact that someone has stolen a bike, is more important than the number of bikes that were stolen.

In the case of access logs, the same reasoning can be used. It is important to see the difference between 0 and 1 hits from a certain area, but the difference between 100 and 101 hits is not very important.

We could also have chosen a function like

$$f(x,y) = \frac{|x-y|}{x+y+1}$$

or any other function that has the required properties. Such a choice must in general be made by a domain expert. We want to emphasise that for the alignment of other multiset sequences, a totally different function might be needed. It all depends on the underlying data represented by the multisets.

Sequences: From actions to multisets

In this subsection we describe how an action sequence is converted into a sequence of multisets, paying special attention to the introduction of so-called *holes* (empty multisets).

Example 2. In this example we again have two sequences $U = \texttt{BAAA}$ and $V = \texttt{BAA}$ of consecutive symbols (actions) from the alphabet $\{\texttt{A}, \texttt{B}\}$. Again, all penalties equal 1. This time, however, each action has a distinct timestamp associated with it: $t(U_0) = 0$, $t(U_1) = 1$, $t(U_2) = 2$, $t(U_3) = 3$, $t(V_0) = 0$, $t(V_1) = 1$ and $t(V_2) = 2$.

A possible alignment is:

```
    0123
U = BAAA
    | ||
V = B-AA
```

The edit distance of this alignment is 1, but the timestamps of the symbols in positions 3 and 4 do not match. When we look at an other alignment like

```
    0123
U = BAAA
    |||
V = BAA-
```

we see that the edit distance itself does not change, but since the timestamps do match in this case, we can say this alignment is better than the first one. □

In general, let $U = U_0U_1 \ldots U_n$ be an ordered sequence of $n+1$ timestamped atomic actions. The timestamps are denoted by $t(U_i)$, where $0 \leq i < n$; so $t(U_0) \leq t(U_1) \leq \ldots \leq t(U_n)$. Let $\Delta t_i = t(U_{i+1}) - t(U_i)$ be the time difference between two consecutive actions ($0 \leq i < n-1$). The first step is to combine actions with (exactly) the same timestamp into multisets. The timestamp of such a multiset is naturally defined as that of one of its elements. So we get an ordered sequence of timestamped multisets.

Now we can still further combine consecutive multisets. This can for instance be done if their timestamps are sufficiently close, or if there is some natural urge to combine them, e.g., into years. In the latter case the new timestamp would be the year, in the former the average could be chosen. However, in any case we assume that the timestamps of all sequences under consideration are from a finite set, e.g., years ranging from 1988 to 2008.

We slightly abuse the notation from the first paragraph, and still use Δt_i as the distance between consecutive multisets from a given sequence. Clearly, in many situations these Δt_i's are not evenly distributed. Sometimes they are, and constitute the same linear range for all sequences under consideration. If this is not the case, there are two natural options. The first is to introduce *holes* (empty multisets) to fill in the "missing" multisets. These holes arise in particular when data is missing or absent, either on purpose or by accident.

If holes at the start or the end of a sequence are created, they can or cannot be omitted — with care. The adding of empty multisets is called *expansion*, and the resulting sequence is called *expanded*. The second option is just to omit these holes, leading to shorter sequences of different lengths. These sequence are referred to as *non-expanded*.

As in Example 1, we have to provide a distance value if a multiset (e.g., a hole in the case of expanded sequences) is aligned to a gap (`-`). Consider, for example, aligning the multiset sequences $\{1,2\},\varnothing,\{3,3\}$ and $\{1,2\},\{3\}$. In order to find an optimal layout, we compute the distance between $\varnothing$ and `-`, and that between $\{3,3\}$ and `-`. We have chosen for a fixed large value, i.e., 1. Note, however, that more intricate possibilities do exist.

7.4 Experiments

We shall now describe two datasets on which we shall apply the techniques described in Section 7.3. The datasets are (unfortunately) not publicly available. However, our focus is on showing the possibilities of our methods and not on efficiency issues. These datasets should be viewed as examples of general databases which contain time stamped bags of atoms.

First we provide a brief explanation of the terms used in this section. There are two types of alignment we use, being *global* and *local* alignment as explained in Section 7.2.

Furthermore there are several options to scale the distances between 0 and 1. We implemented two, namely an *absolute* one, which uses a constant (usually the length of the largest sequence in the database) for scaling. Another option is to use *relative* scaling, where we use the length of the largest sequence in the pairwise alignment to scale the distances.

Finally we implemented two ways to deal with the absence of data. As explained in Section 7.3, we can have *expanded* and *non-expanded* sequences.

7.4.1 Criminal careers

The first database consists of approximately one million (anonymised) criminals and their crimes grouped per year. In this particular case, we only know the number of crimes in certain categories per criminal per year. The number of categories is nine.

Example 3. In Table 7.1 we see a table of two criminals and the crimes they committed over the years.

	1999	2000	2001	2002	2003
A	$\{1,2\}$	$\{3\}$	$\{1,1,3\}$	$\{2,3\}$	$\{3\}$
B	$\{3,3\}$	$\varnothing$	$\{3,4\}$	$\{3,3\}$	$\{3,4\}$

Table 7.1: Two criminal careers.

Per criminal, each year can be viewed as a multiset of crimes (1, 2, 3 and 4 are crimes in this example) and the entire criminal career is a sequence of multisets. The second criminal has a hole for the year 2000. □

In order to do analysis on the behaviour of these criminals, we use the previously defined distance measure for sequences of multisets. With this, we can identify trends in criminal behaviour and try to extrapolate future behaviour.

As mentioned before, we try different combinations of alignment, scaling and treatment of holes. To illustrate the difference between expanded and non-expanded sequences, we first give an example.

Example 4. Suppose we have a criminal that has committed the following crimes:

year	1998	1999	2002	2003	2004
crimes	$\{1,2\}$	$\{3,3\}$	$\{2\}$	$\{1,2\}$	$\{2,3\}$

Table 7.2: Example criminal.

We can now use the non-expanded sequence $(\{1,2\},\{3,3\},\{2\},\{1,2\},\{2,3\})$. The years 2000 and 2001 in which the perpetrator was inactive, are simply ignored. The only thing that is preserved is the order in which the multisets of crimes are committed.

We can also use the expanded sequence $(\{1,2\},\{3,3\},\varnothing,\varnothing,\{2\},\{1,2\},\{2,3\})$ for our alignment. The years where there was no activity are explicitly denoted as empty multisets. This way both the order and the time aspect are preserved. □

Depending on the type of analysis, one of these choices is to be preferred. In the case of criminal careers, a hole in the activities may be the result of a sentence (jail). Depending on various assumptions we may or may not want to include the holes in our analysis. In this section we shall see the difference between both choices. Furthermore, no weighing has been applied (meaning that each type of crime is considered equal in severity).

The pictures below are obtained by a dimension reduction algorithm comparable to Multi Dimensional Scaling [5]. This technique iterates over all pairs of data points and adjusts the position of these points on a 2-dimensional plane according to the desired distance between them. It is a kind of competitive neural network which is halted when the position of the points no longer change. The output of this algorithm is an embedding of the points in a 2-dimensional *torus* (see Chapter 3). The usage of a torus improves the embedding for data that is non-flat (i.e., can not be embedded in a normal 2-dimensional plane). So one should be aware that the boundaries of the pictures are identified and that the maximum distance in each of these pictures is actually half of the diagonal. We remark that we just use this technique to give an impression of the results obtained by the usage of our metric. The distances are of importance,

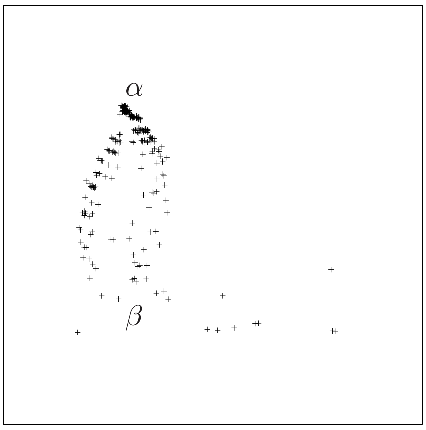

Figure 7.1: Database of criminal careers. Global alignment, absolute scaling, non-expanded careers.

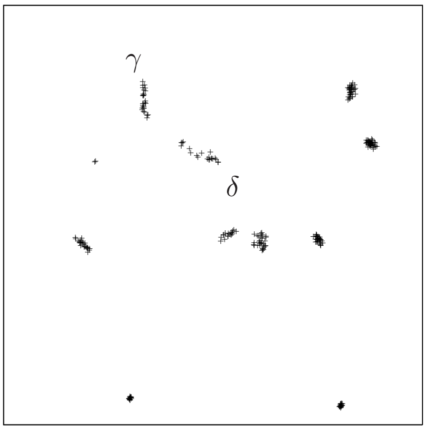

Figure 7.2: Database of criminal careers. Global alignment, relative scaling, non-expanded careers.

not the dimension reduction technique itself. Using other dimension reduction techniques will provide similar results.

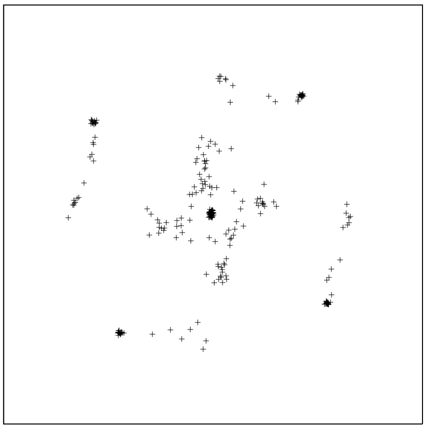

Figure 7.3: Database of criminal careers. Local alignment, absolute scaling, non-expanded careers.

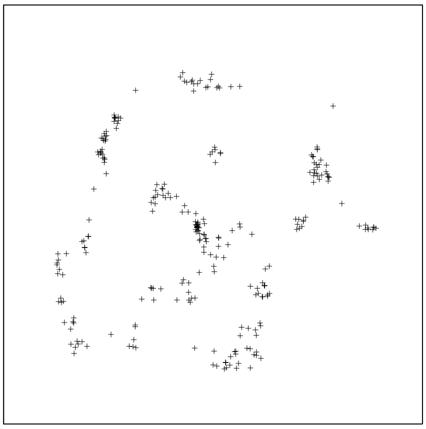

Figure 7.4: Database of criminal careers. Local alignment, relative scaling, non-expanded careers.

In Figures 7.1–7.4 we see a visualisation of 5,000 criminals; all pictures were made with non-expanded careers. In Figure 7.1 we used a global alignment and an absolute scaling factor. In Figure 7.2 we used a global alignment with a relative scaling. Figure 7.3 is made with local alignment and an absolute scaling factor and Figure 7.4 is made with local alignment and a relative scaling.

In the first two of these pictures, there is an emphasis on the length of the careers, which in this case means the amount of years in which a criminal has been active (since all non-active years are ignored). In Figure 7.1 for example, we see a large cluster indicated with the letter α, these are all short careers. As we move from α to β the length of the careers increases.

In Figure 7.2 the clusters indicated with γ and δ contain only long careers. The other clusters have careers of roughly the same size and their elements have been clustered according to the similarity in career.

In the other two pictures, we see something different emerging: the long careers are no longer put into the same cluster, but more emphasis is given to similarity in criminal activity. For example, the central cluster in Figure 7.3 consists mainly of criminals who have committed non-violent crimes involving large sums of money.

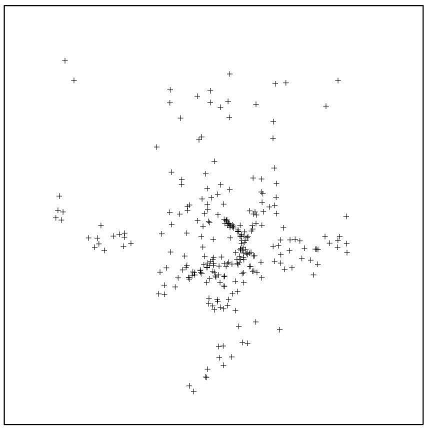

Figure 7.5: Database of criminal careers. Local alignment, relative scaling, expanded careers.

When we use expanded careers as in Figure 7.5, we see a far more scattered image than in the previous ones. The reason for this is that the number of dimensions increases, combined with the fact that the gap penalty is constant. We see that a series of holes in a career will result in a rather big distance, whereas this is not the case with non-expanded careers. On an individual basis, this technique can be useful because it is able to detect an overlap in imprisonment, the clusters, however, will be very small (perhaps indicating criminals who work closely together).

7.4.2 Access logs

As a second dataset we used the access log of a web server. For confidentiality reasons we can not disclose the actual dataset. We looked at the class-A network (i.e., the first number of the IP address) from which a hit in this log originated.

Within a timeframe of sufficient length, these hits will result in a multiset since more than one hit can originate from a specific network. Note that events with different timestamps will be combined into multisets, and that usually holes do not occur — except for periods when the web server was inaccessible, for example.

A sequence of these multisets can be defined in multiple ways. First of all, it is natural to choose a timeframe of constant length. For the choice of the consecutive timeframe however, there are many possibilities. One can choose to have an overlap in the timeframes; this will result in sequences having an overlap by design. Especially for the analysis of access logs, this is a good choice, since when visualising the data, the consecutive sequences will be placed next to each other, giving a very natural view.

Of course one can vary the overlap; this will result in visualising different aspects of the data. A large overlap will result in highly rigid strains of sequences, which only a great similarity with other sequences can break. Decreasing the overlap will result in more loose chains, and more emphasis is given to similarity between sequences based upon their content instead of the chronological order.

In our experiment, we were interested in the class-A network from which a hit in the log originated. We grouped all hits within 10 minutes together into a multiset and made a sequence of multisets of length 10. The subsequent sequence starts 10 minutes after the first one, therefore having an overlap of at least 80% (only the first and last multiset can differ from its predecessor).

For this experiment we took the first twelve hours of the log and another part of twelve hours, exactly one week later. Since all sequences in this test set are of the same length, there is no difference between local and global alignment, there is also no difference between absolute and relative scaling and finally, there is no difference between expanded and non-expanded careers, since in the log there are hardly any time windows of 10 minutes where no activity occurs.

In Figure 7.6 we see that a lot of consecutive sequences (denoted by numbers in this picture) are very close to each other and are neatly arranged in ordered threads. The sequences denoted by the numbers 26 to 31 and 101 to 108 are prominent examples. This behaviour occurs throughout the picture, but is perhaps not clearly visible because of the overlapping numbers.

In Figure 7.7 we see exactly the same visualisation, but we have replaced the numbers with +s for a clearer view. Again, we see the ordered threads, but now we see that some of these threads align next to each other and sometimes even intersect. This indicates a similar behaviour in these consecutive sequences.

We also expected a correlation between sequences one week apart, but except for the pair $(8, 86)$ (in the top cluster of Figure 7.7), no evidence of this could be found.

7.5 Conclusions and further research

We have introduced a new metric that is a natural way of defining a distance between sequences of multisets. We have used the example of criminal careers

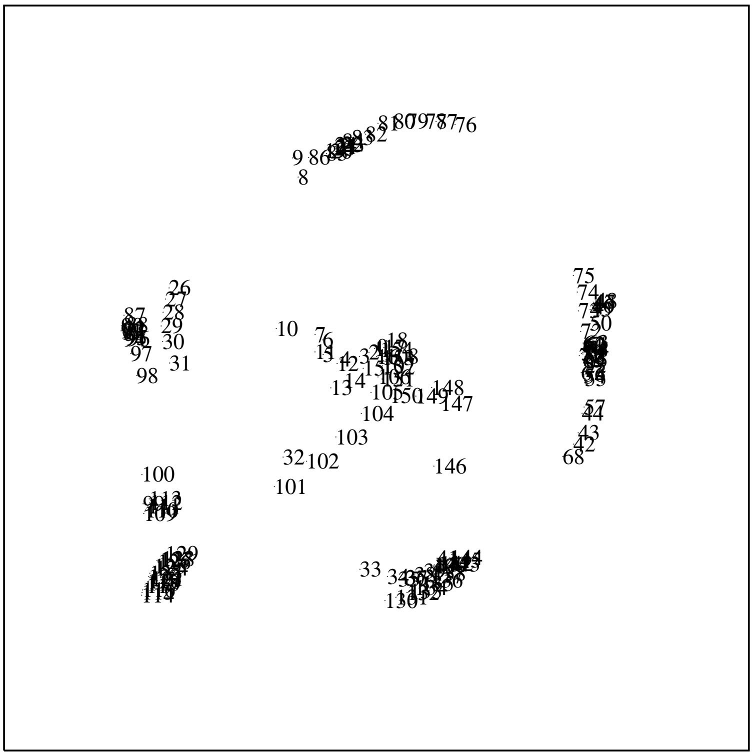

Figure 7.6: Access log, two periods of 12 hours, 1 week apart.

to illustrate this point and to emphasise the flexibility of this new metric. As a second example, we used access logs to show that our metric can be applied in a different domain. We used a visualisation technique to show the merits and possibilities of the method, highlighting several user-controlled features.

In practice, we often do not have real valued timestamps. Transactions are usually grouped together in days, months or even years. This can simplify alignment, since absence of data can be viewed as an empty transaction (one with no elements in it). Alignment of this type of sequences is straightforward.

A useful extension on alignment is the use of variable gap length (also see [47]). In many practical cases, the fact that there is a gap in the alignment is more important than the length of the gap. As long as the gap penalty increases with the length of the gap, and as long as the function governing the gap penalty is concave, the triangle inequality of the edit distance holds. Unfortunately, the efficiency of the alignment algorithm deteriorates. Because there is more data dependency, the time complexity will be $\mathcal{O}(n^3)$ and the space complexity will increase to $\mathcal{O}(n^2)$. Even when working with discrete timestamps, one could consider less penalty for consecutive empty transactions.

We also would like to investigate the usage of weights for the crimes. At

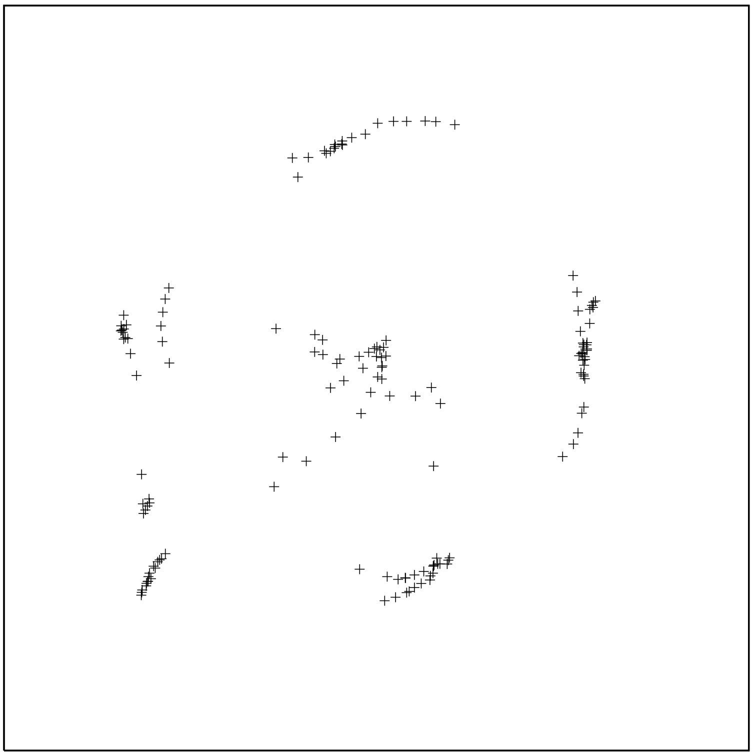

Figure 7.7: Same as Figure 7.6, with numbers replaced with +'s.

present, we have not done so because there is no clear weighing scheme for the categories available yet.