"Improving sequence analysis for the social sciences: a new and more useful method to determine similarity between sociological sequences"

Abstract

Sequence analysis has been an increasingly popular tool to find patterns in sociological sequences. Sequence analysis compares sequences individually on similarity after which similar sequences are clustered into distinct groups. Analysing how and why certain groups are different from other groups yields important insights.

This paper proposes more useful method to calculate sociologically valid similarity values between sequences.

It is shown that the proposed method does not only yield sociologically expected results, it also outperforms existing algorithms with regard to the valuation of order and the support for time-dependent substitution matrices. Moreover, it supports both single-channel and multi-channel sequences, as well as sequences of unequal length. Finally, tests on existing data-sets show that the new method produces sociologically expected results for real-life data, and that the algorithm is confident in doing so.

Date

August, 2011

Author

Name:	Willem Mulder
Student number:	3183300

Context

Faculty:	Geosciences
Programme:	Science and Innovation Management (SIM)
ECTS:	45
Supervisors:	Dr. Andrea M. Herrmann, Dr. ir. Alexander Peine

Preface

The method as found in this paper was originally created to help analyse sequences of entrepreneurial start-up activities: existing methods for analysing those sequences were rather limited, especially when activities happened simultaneously, and so I created a new method to improve on these areas. While working on the new method however, it became clear that the project would be beneficial for more than just the initial entrepreneurial research, and thus over time, the scope of this research widened. It went from a specific entrepreneurial subject on start-up activities, to entrepreneurial research in general and finally to the social sciences in general.

Still, it was the first entrepreneurial research on start-up activities that lead this paper from start to end and made it the way it is. I would like to thank the creator of that entrepreneurial research, Dr. Andrea M. Herrmann to be the inspiration of my project, and would like to thank her even more for being my supervisor during this master thesis project. If it was not for her research with regard to sequence analysis, I would have never started to even think about analysing sequences or about how it could be improved.

Additionally, it was her that introduced me to Anette Fasang, who has helped me with literature, good discussions and constructive criticism. Thanks Anette!

Finally there was family, friends and my soon to-be wife Lona that listened to my stories, endured my extensive explanations of complicated algorithms and gave me feedback and thoughts throughout the process.

It was for all of you, and the help of God, that I finished this paper! Thanks again.

Table of Contents

1. Introduction :

how Sequence Analysis is useful but limited in its current state

Recently, many sociologist researchers have been showing increased interest for the method of sequence analysis (e.g. Garnsey et al., 2003; Brzinsky-Fay, 2007; Massoni et al., 2009), with which they are able to analyse sequential data in a quantitative manner. With the ever increasing power of computers, Sequence Analysis (SA) has become a popular method for doing social research in a wide variety of topics ranging from originally finding patterns in dance-steps (Abbott and Forrest, 1986) to more recent studies on innovation (Van de Ven and Poole, 1995), family histories (Elzinga and Liefbroer, 2007), careers (Scherer, 2001) and life-course trajectories (Wigginset al. 2007; Martin, Schoon, and Ross 2008).

In all these studies, SA helps sociologists to find distinct groups within a larger set of sequential observations (Abbott and Forrest, 1986; Abbott and Tsay, 2000; McIndoe and Abbott, 2004). This is of much interest to sociologists, since the analysis of groups, and specifically how and why certain groups are different from others, is a significant field of work within the social sciences.

Still, despite the usefulness of SA for sociologists to find distinct groups within a large set of coded sequences, there was also critique on the exact implementation of SA (Wu, 2000; Levine, 2000), specifically with regard to the Optimal Matching (OM) method that is used to determine which sequences are similar[1] to each other, and which are not.

OM originated from the field of biology where it was used to analyse sequences of DNA nucleotides on similarity (Needleman and Wunsch, 1970). It was only later that Abbott and Forrest (1986) introduced the method to the social sciences, where it slowly gained popularity, and finally became the most common method to determine similarity-values between sociological sequences (Elzinga, 2003). Wu (2000) and Levine (2000) however criticized the biological origin of the OM method and even questioned the validity of the OM algorithm for the field of the social sciences. An evolving debate lead journals to publish a multitude of individual articles on the subject[2], spurring not only discussion but also adaptations to the original OM method to make it more useful for use in the social sciences. Over the years, OM has seen many improvements and the amount of papers that use OM for doing analysis has been increasing ever since.

1 Some authors originally use 'dissimilarity' or 'distance' to describe relations between sequences. However, as Elzinga (2003) notes, it is better to use 'similarity' instead, since we want to group sequences on similarity, not on dissimilarity.

2 For example Abbott (2000), Abbott and Tsay (2000), Gauthier et al. (2009), Halpin (2009), Hollister (2009), Lesnard (2010), Aisenbray and Fasang (2010), Brzinsky-Fay and Kohler (2010) and Gauthier et al. (2010).

Still, despite the progress and many useful contributions over the years, some of the early critique still holds, and it is evident that the OM algorithm has several major limitations (Abbott and Tsay, 2000; Elzinga, 2003; Hollister, 2009; Gauthier et al., 2010). This is mainly due to three issues which in short[3] are primarily concerned with the following.

First, and most importantly, the OM algorithm has difficulties yielding sociologically valid results when using multi-channel data, i.e. when using data that contains simultaneous events (Abbott and Tsay, 2000; Gauthier et al., 2010). Considering the strong agreement in the literature regarding the importance of multidimensionality in the analysis of life trajectories[4], the lack of proper support of multi-channel analysis in OM is a serious shortcoming (Abbott and Tsay, 2000). Indeed, the OM algorithm was originally written to analyse DNA sequences that only ever have one element at a time, and is unable to calculate distance-values between sequences that have multiple channels. Over the years, researchers used an improvised solution to force multichannel sequences into unilinear sequences using cross-classifying elements (Abbott and Tsay, 2000), but in the meantime it was acknowledged that using such a method would discard much of the richness of data and would cause problems for data-analysis due to the exploding element-alphabet (Han and Moen, 1999; Gauthier et al., 2010). A recent proposal for multi-channel analysis by Gauthier et al. (2010) can be useful for specific cases[3], while additionally this paper shows that its results are sociologically unexpected in a multitude of cases. Consequently, the social sciences still lack a sociologically valid multi-channel SA method.

A second limitation is that the internal workings of the OM algorithm lead it to undervalue identical order between sequences[5] (Dijkstra and Taris, 1995; Elzinga, 2003). Instead, it favours identical length and identical tokens. This implies that sequences of different length that share the very same order among their tokens can be considered by OM to only have very low similarity, while sociologically they would be considered quite similar due to having the same order but only with a slightly different timing.

Third and finally, the OM algorithm treats specific differences between sequence elements as equal for any time-position in the sequence, while sociologically the impact of differences can severely vary depending on the context (Hollister, 2009; Lesnard, 2010; Halpin, 2010). Indeed, the sociological meaning of a difference often varies over time, depending on how social groups interpret and evaluate the difference

3 See Chapter 2 for a more detailed discussion.

4 See Gauthier et al. (2010) for an overview. Examples include Elder (1985), Clausen (1986), Kohli (1986), Levy (1991, 1996), Giele and Elder (1998), Heinz and Marshall (2003), Mortimer and Shanahan (2003), Levy et al. (2005), Macmillan (2005).

5 It should be noted that one advantage of OM over other algorithms such as the Hamming Distance is the very fact that it at least partially takes order into account. However, the notion of order in OM is implicit and it has only a marginal impact on sequence similarity.

at a certain time. Thus, (dis)similarity between sequence elements should be time-dependent. Lesnard (2010) has proposed a solution to this problem, but it is limited in that it only works for sequences of equal length.

Above problems of the OM algorithm severely limit the usefulness of SA in several cases: OM not only ignores contextual richness of data and undervalues order, it moreover fails to work with the simultaneous nature of reality. One might wonder whether OM is the one and ideal solution for the social sciences to calculate sequence similarities, and indeed it is exactly that question that led Dijkstra and Taris (1995) and Elzinga (2003; 2007) to create alternative methods.

Yet, these methods come with their own problems. First, the method of Dijkstra and Taris (1995) is a variant on OM that inherits the same problems of multichannel data and contextual differences, and is additionally only useful for sequences without repetitions (Abbott, 1995; van Driel and Oosterveld, 2001; Elzinga, 2003). Second, while the subsequence method of Elzinga (2003; 2007) is specifically good at taking order into account by focussing on "the basic properties of a sequence: its constituting elements and the order in which they appear " (Elzinga, 2003, pp. 6), it still fails to work with multichannel data and does not allow one to use substitution matrices of any kind. The latter lack of substitution matrices is a serious limitation, because substitution matrices are the only way for researchers to specify which sequence-tokens are to be seen as similar or dissimilar[6]. In discussion of Elzinga's method, literature argues that "given the widespread use of variable substitution costs in sociological applications, [the lack of support for substitution matrices] is a serious limitation" (Hollister, 2009, pp. 242).

All in all, while many years of scientific development have significantly improved upon the original OM method and have led to many new findings, it is easily concluded that the social sciences still lack a generally usable, sociologically valid method to calculate similarities between sociological sequences. Acknowledging the importance of such a method for the social sciences, this paper proposes the 'Mulder algorithm'[7] that tries best to avoid the drawbacks of OM and subsequently aims to help filling the gap that has been evident for some time now in the sociological literature, specifically with regard to support for time-dependent differences, the valuation of order and most importantly the general support for multi-channel analysis.

To make explicit that the Mulder algorithm should be more useful than existing algorithms, the specific question that this paper aims to answer is "To what extent is using the Mulder algorithm in an SA analysis more useful for yielding sociologically valid (dis)similarity values than using existing algorithms?"

6 Indeed, considering that the tokens represent activities or socio-economic statuses, it is evident that some activities or statuses can be sociologically very similar (e.g. "having 4 children" or "having 5 children"), while others are sociologically very dissimilar (e.g. "having a child" and "being unemployed").

7 Simply referring to the author.

Obviously, 'useful' is a subjective term. This paper interprets usefulness as the ability to be used for a certain purpose, in this case the purpose of using available data (i.e. single-channel and multi-channel sequences as well as time-dependent similarity) to yield sociologically valid (dis)similarity values for pairs of sequences. 'Sociologically valid' results are exactly those results that literature and people in general would consider to be fitting for the input data. Consequently, exactly the ability of the algorithm to produce results that are sociologically expected with regard to the available input data, make the algorithm useful.

This paper shows that the Mulder algorithm is more useful than existing algorithms in four ways. First, the Mulder algorithm is able to yield sociologically valid result for multi-channel data, something that existing algorithms only do partially, be it because current algorithms do not support multi-channel data at all or because they neglect much of the social richness of data, resulting in sociologically unexpected results. Second, the Mulder algorithm can differentiate element-similarity over time, using time-dependent substitution matrices. Currently these time-dependent substitution matrices are only supported by a very simple algorithm that additionally can only analyse sequences of equal length. Third, it is shown that the Mulder algorithm values order as sociologically expected for both single-channel and multi-channel data. Currently, there is no algorithm that is able to analyse multi-channel data while properly valuing order. Fourth, when using the (dis)similarity-values from the Mulder algorithm to create clusters for larger data-sets, these clusters are more distinct than when using results from existing algorithms, suggesting that the Mulder algorithm is better able to determine which sequences are similar and which are not. Objective measures for cluster quality confirm this outcome.

To illustrate these points, the paper continues as follows. Chapter 2 gives an overview of the developments in the field of Sequence Analysis. Specifically, it highlights how literature has been considering certain functionality to be important for SA, both regarding the mere ability to analyse certain kinds of data as well as how certain data should yield specific sociologically valid results. Chapter 3 gives a succinct overview of the desired functionality and additionally shows how existing SA algorithms do only partially provide this functionality. Chapter 4 introduces the Mulder algorithm and uses theoretical reasoning to show how the Mulder algorithm provides the desired functionality and avoids the drawbacks as evident in current algorithms. The chapter starts from a simple idea and then continues to develop this into the more complex Mulder algorithm that provides more functionality than any current algorithm does. Three different implementations of the Mulder algorithm are presented. Chapter 5 then provides an in-depth explanation of the internal workings of the Mulder algorithm. Chapter 6 validates the Mulder algorithm by formally testing whether the Mulder algorithm provides the functionality as discussed in chapter 3 and 4. Additionally, it benchmarks the different implementations of the Mulder algorithm, as to conclude on the most optimal one. Chapter 7 then uses this optimal implementation of the Mulder algorithm to provide a more in-depth comparison between the Mulder algorithm and existing algorithms to show that the Mulder algorithm is more useful in providing sociological valid (dis)similarity values than existing algorithms. A discussion and conclusion review the process and results, while providing recommendations for future research.

Note that this paper does not pretend to provide a distance-calculating algorithm that is superior in a once-and-for-good fashion. It is instead meant to illustrate that a new algorithm could solve some of the problems that are currently evident in SA. The Mulder algorithm serves as a possible step towards vastly improved methods for SA in the social sciences, and it is hopefully through constructive scientific discussions that many more contributions will be made to achieve this goal.

2. Developments in Sequence Analysis :

from its first introduction to the latest expectations for the method

In order to answer the research question and create a new method of sequence analysis, it is important to be clear about the sociological consensus and expectations with regard to SA as they have developed over the years. Mainly, this includes expectations with regard to how SA should be able to deal with available data and about how certain data should lead to specific results. Only once these expectations are known it is possible to create a method that solves evident problems while not falling short on other functions and possibilities that the social sciences have become to expect of SA.

The history of SA starts more than three centuries ago, in the biological field. Biologists needed a method to analyse how different a DNA sequence was from other sequences. Doing this by hand is tedious and subject to arbitrary choices, so it was Needleman and Wunsch (1970) that first published an objective algorithm to quantitatively assess the problem, later followed by an improved algorithm by Sankoff (1972). The method works by taking two single-channel sequences of amino-acids for which it computes the minimal amount of necessary mutations to transform the one sequence into the other (Needleman and Wunsch, 1970). It is important to note that the method was created to mimic nature and so the available mutations are those that occur in real life. The first type of mutations is an 'indel' operation with which exactly one element is added to or removed from a sequence[8]. The second type is the substitution operation with which exactly one element of a sequence is substituted for

8 Note that when comparing sequences and counting the amount of mutations necessary to transform one sequence into the other, it does not matter if one adds an element to the first sequence or removes it from the second. In both cases the sequences will be dissimilar by one operation: the very same indel operation.

another. Any such operation necessary to transform one sequence into the other represents a distance between the investigated sequences. Needleman and Wunsch (1970) acknowledged that not all operations contribute equally to this distance. In general, substitutions cost more than indels and a rare substitution can cost more than a more common substitution. The costs of the various substitutions can thus be varied and can be stored in a substitution matrix that holds the substitution costs for any pair of element-types. While calculating the distance-value between two sequences, the necessary indel and substitution operations are multiplied by their respective costs to yield a final distance. This final distance is always optimal in the sense that it is the result of exactly that set of operations that yields the the smallest overall costs. Indeed, there are infinite ways to transform one sequence into another, but only the most optimal is chosen. It is due to this optimal behaviour that the the algorithm was labelled Optimal Matching (OM) when it was introduced to the social sciences by Abbott and Forrest (1986).

The reason to introduce SA to the social sciences was its distinctive feature to calculate distances between sociological sequences in a quantitative manner (McIndoe and Abbott, 2004). Previously, the problem where "one wants to find a characteristic sequence, or several characteristic sequences, among a large sample of sequences [and] seeks to explain variation among these sequences in terms of some external variables" (Abbott and Forrest, 1986, pp. 474) had to be solved using time-intensive qualitative analysis "using an 'ideal type' or comparative analysis, and then to consider the variations from it on an individual basis" (ibid, pp. 471). The quantitative method of SA turned this intensive work into a matter of distinct calculations that could be executed on computers. Similar to the sequences of amino-acids, Abbott and Forrest (1986) used the SA algorithm to analyse uni-linear strings of data where observed social events or socio-economic statuses are ordered in time, but do not occur simultaneously.

The social sciences now had access to an algorithm that could quantitatively calculate distances between uni-linear sociological sequences. The substitution matrices as used in biology were used to set costs for the different substitutions of social events or socio-economic statuses for others. Early applications merely used theoretical grounds to set those matrices. Events or statuses were classified into tree-like structures so that "each pair of figures can be characterized by the number of steps up the hierarchy necessary to put them under a common heading. The ratio of this number to the total number of steps possible, [...] is the cost of substitution " (Abbott and Forrest, 1986, pp. 479). Gaps in sequences were also accounted for in the sense that an indel or substitution of a gap would simply have a certain cost.

An evident problem however when using the absolute distance-values as generated by the OM algorithm was that distance-values tend to become incomparable since a distance of e.g. 2 does not, ultimately, say anything about how much two sequences are similar. Indeed, a distance of 2 on a pair of two very large sequences is insignificant, but a distance of 2 on a pair of very short sequences is substantial. Abbott and Tsay (2000, pp. 13) acknowledged this problem and note that "variation in sequence length means that some pairs of sequences have a greater potential distance between them than do others. [Many] have dealt with this by the expedient of dividing the ultimate cost of transformation by the length of the longer sequence of the pair" . This method can be easily disputed by arguing that potential distance for a pair is not just dependent on the longer sequence, and indeed Elzinga (2007) proposed to use the geometrical mean of the length of the two sequences, but the most important thing is that resulting dis(similarity) values should be relative rather than absolute. Ideally, the values range between 0 and 1 to enable for interpretation in terms of percentages.

It was maybe due to the above and other problems that, despite the widespread use of SA in the biological field (Abbott and Tsay, 2000), researchers in the social sciences did not pick up SA so quickly. It was mainly one of the two original proposers of SA for the social sciences, Abbott, who used the method in several articles (e.g. Abbott and Hrycak, 1990; Abbott, 1995; Abbott and Tsay, 2000) to demonstrate the usefulness of SA and OM. It was only more than ten years after the initial article by Abbot and Forrest (1986) that scientists from the sociological field started to study SA more, and it was in August 2000 that a special issue on SA in Sociological Methods & Research introduced the 'first wave' in SA development.

Discussions in the first wave were mainly concerned with the sociological validity and applicability of the OM method (Wu, 2000; Levine, 2000; Abbot and Tsay, 2000) and more practically, the question how to deal with timing and order (Dijkstra and Taris, 1995; Wu, 2000; Elzinga, 2003). The latter problem of order was already acknowledged by Dijkstra and Taris (1995) when they noticed that the OM algorithm undervalued order, and consequently introduced a variation on the OM method to specifically take order into account. While the method has been criticized for only being useful for sequences without repetitions (Abbot, 1995; Elzinga, 2003), the paper (Dijkstra and Taris, 1995) presented a fundamental list of how distance-calculating algorithms in SA should take specific properties of sequence-pairs into account when determining similarity. Elzinga (2005) used and expanded the list and presented the premises as follows.

1. Sequences that have no elements in common are maximally dissimilar.
2. Sequences that have the same spells in the same order are maximally similar.
3. The more elements that sequences have in common, the more similar these sequences are.
4. The more common order there is among common elements, the more similar sequences are.

Note that the list explicitly includes the notion of order, which the OM method tends to undervalue in its calculations (Dijkstra and Taris, 1995; Elzinga, 2003).

It is this notion of order that was an important element for Elzinga (2003) when he created an algorithm that "relies on the basic properties of a sequence: its constituting elements and the order in which they appear" (Elzinga, 2003, pp. 7). Still, while the different similarity-measures that Elzinga proposed (Elzinga, 2003; Elzinga, 2005; Elzinga, 2007) deal with order very well, the algorithm remained a small sub-field within SA due to the inability to support the substitution matrices (Hollister, 2009), the latter being regarded as an elementary part of SA analysis (ibid).

It was then in 2009 that a second wave of SA-related papers emerged (Gauthier et al., 2009; Halpin, 2009; Hollister, 2009; Lesnard, 2010; Aisenbray and Fasang, 2010; Brzinsky-Fay and Kohler, 2010; Gauthier et al., 2010). During this second wave, SA had been accepted as a general pattern-finding method, and discussions proceeded to identify specific limitations with regard to OM as well as to propose solutions to help SA advance and solve the discussed problems (Aisenbrey and Fasang, 2010).

Three developments are noteworthy when discussing how literature expects certain functionality of SA and OM.

First, the use of substitution matrices has become common practice (Hollister, 2009). The ways in which the substitution matrices are generated have also become more elaborate, ranging from the old method of theoretically or emperically deriving values to substitution matrices that are generated automatically from the data, using transition-rates or other measures. Gauthier et al. (2009) and Hollister (2009) give an extensive overview of the different possibilities. In addition to the use of one substitution-matrix for all time-positions, there has been a call to have different substitution matrices for different absolute time-positions (Lesnard, 2010; Aisenbrey and Fasang, 2010). The suggestion of multiple substitution matrices is based on the idea that specific differences between sequences can be substantial in some era but not in others (see figure 1 for an example). To vary the (dis)similarity between certain elements, it should be allowed to have different substitution matrices for different absolute time-positions[9]. To illustrate the importance of having this feature, I cite Aisenbrey and Fasang (2010, pp. 451) who describe time-dependent substitution matrices as the most important area for improvement in SA and argue that "the elaboration of time-dependent cost schemes is critical".

9 Note that the OM algorithm cannot use this feature simultaneously with the use of indels. Indels warp time (i.e. they shift parts of a sequence to the left or right) so that it is impossible to determine substitution matrices for a specific time-position. Consequently, Lesnard (2010) refrains from using indels and creates a substitution-only algorithm (i.e. a Hamming Distance) named Dynamic Hamming Matching (Lesnard, 2010).

figure 1: example to illustrate Lesnard (2010)

Consider two eras. Era 1 in which the sociological status for having children is equal to the sociological status of *not* having children. Era 2 in which having children is seen as very important.

Now consider the following sequences where A means having children and B means having no children.

Era	Sequence 1	Sequence 2	Sociological difference
1	AAAA	BBBB	Very small difference. A and B are equal.
2	AAAA	BBBB	Very high difference. Having children is very different from not having children.

It is beneficial to have different substitution matrices for the different eras. In practice, this would demand that every time-position has its own substitution-matrix.

A second issue discussed during the second wave of SA is how the similarity between an element in one sequence and an element in another sequence is dependent on the context of those elements in their respective sequence (Hollister, 2009; Halpin, 2010). An indel that extends a certain spell of elements with one is much less significant than the insertion of a very uncommon element in the middle of the spell. See figure 2 for an example. Hollister (2010, pp. 246) concludes that "the traditional fixed-cost indel system [...] creates what I call the free ride problem: Once the decision is made to insert or delete an element, the OM algorithm doesn't care what element is inserted. It allows a highly unusual element to be inserted without any additional cost". It would be useful to determine the similarity between sequences not only by a set of (non)matching elements, but also by the context in which these matches or nonmatches occur. For the OM algorithm, this has been implemented by taking into account the adjacent elements for an indel (Localized OM, see Hollister, 2009) or the length of the overarching spell for adjusting substitution or deletion costs (OMv, see Halpin, 2010).

figure 2 : example to illustrate Hollister (2009) and Halpin (2010)

Consider two pairs of sequences. Each pair consists out of one sequence that is a spell of elements, and another sequence that is the same spell, but with an element A added in the middle of the sequence.

Pair	Sequence 1	Sequence 2	Sociological difference
1	BBBB	BBABB	High. Sequence 1 is broken in two.
2	AAAA	AAAAA	Lower. Sequence 1 is extended by one element.

It can be seen that the sociological meaning of the addition of the A can differ for any pair, depending on the context in which the A is added.

The third important issue discussed during the second wave of SA is how distance-calculating algorithms should deal with sequences that contain simultaneous elements (Gauthier et al., 2010). The OM algorithm originally was created to deal with single strings of amino-acids (Needleman and Wunsch, 1970) and consequently the OM algorithm expected elements strictly after each other, not simultaneous to each other. Ever since the OM method was ported from the biological field (Abbott and Forrest, 1986), it was clear that the single-channel nature of OM was a severe limitation for SA (Abbott and Tsay, 2000) and that SA somehow needed to integrate the possibility to analyse multichannel sequences. Indeed, literature has long stressed the importance of multidimensionality in the analysis of life trajectories[10] and it is evident that most activities or socio-economic statuses are not mutually exclusive, leading to sequences with simultaneous elements. Researchers that were confronted with multi-channel data but a single-channel OM method (e.g. Abbott and DeVinney, 1992; Dijkstra and Taris, 1995; Blair- Loy, 1999; Aassve et al., 2007; Pollock, 2007) went to recode their data to make it fit for the OM method. Abbott and Tsay (2000, pp. 8) recall that "a number of authors have created complex 'events' for their sequences by cross-classifying a number of simple events. This is the only strategy available for dealing with multiple, parallel tracks of sequence information in the OM framework; it must be reduced somehow to the unilinear structure expected by the OM algorithms". The strategy of combining events does however take away largely from the longitudinal information and temporal interdependence of elements that are contained in sequence data (Gauthier et al., 2010) while also creating uncomfortably large element-alphabets (Han and Moen, 1999; Gauthier et al., 2010) for which it is additionally very difficult to create substitution matrices (Gauthier et al., 2010). The new MCSA method proposed by Gauthier et al. (2010) avoids the combination of different channels into one but instead executes a separate distance-calculation for every channel, after which the results are averaged. This approach however has its limitations. First, it requires separate channels with multiple mutually exclusive tokens, which makes this method only useful in specific cases. Second, the inter-channel interactions are not taken into account so that much of the multi-channel richness of data is lost. Thus, a general matching method that can deal with simultaneous tokens while leaving intact the full multi-channel richness of the data, has not yet been proposed, and the alternative methods of Dijkstra and Taris (1995) and Elzinga (2003; 2007) do not provide multi-channel functionality either. Creating a method that is capable of analysing simultaneous statuses or events would be of great value for social scientists to extend their analytical possibilities into the analysis of multi-channel sequences.

All in all, while sequence analysis has shown to be a very useful tool to analyse sequential data in a quantitative manner, there are some standing limitations with

10 See Gauthier et al. (2010) for an overview. Examples include Elder (1985), Clausen (1986), Kohli (1986), Levy (1991, 1996), Giele and Elder (1998), Heinz and Marshall (2003), Mortimer and Shanahan (2003), Levy et al. (2005), Macmillan (2005).

regard to how SA can do what researchers in the social sciences would like it to. These limitations are concerned with how the current distance-calculating algorithms most importantly can not analyse multi-channel sequences and additionally either cannot take order very well into account (OM), cannot analyse sequences with repetitions (Dijkstra and Taris method) or cannot use substitution matrices (Elzinga method).

Thus, this paper proposes a new method to try and overcome these limitations by providing a method that works with both single-channel and multi-channel data, while yielding sociological meaningful similarity-values (i.e. explicitly considering order and context) and being able to take into account theoretical knowledge about the data as provided by the researcher (i.e. support substitution matrices).

Since the proposed method is new, it is important to not only overcome current limitations but to also include all the functionality of the OM algorithm that sociologists have come to expect of a distance-calculating algorithm. The next chapter aims to give an overview of desired functionality, and in how far current algorithms provide this functionality.

3. Desired functionality :

About how the literature wishes SA to have certain functionality and about the extent to which current algorithms provide it

It can be seen that over the years, the social sciences have been aiming to have a method of sequence analysis that is able to calculate relative similarity between pairs of sequences while being able to do some specific things. This chapter gives a concise overview of the expected and desired functionality of a distance-calculating algorithm, as found in the literature that is briefly discussed in the previous chapter.

In addition, researchers should be allowed to vary the way the algorithm looks for patterns. Indeed, it has always been the idea that "Sequence Analysis looks for patterns or regularities. The type of regularity can be varied by varying the structure and parameters of the algorithm." (Abbott, 2000, pp. 67). Yet, OM does not allow for varying structure or parameters of the algorithm to gain new insights with the same data, beyond using substitution matrices. It could be an interesting addition to SA to allow sociologists to specify which similarities or types of similarities they deem important.

1 First of all, the algorithm should adhere to the principles as proposed by Dijkstra and Taris (1995) and stressed repeatedly by Elzinga (2003; 2005), concerning how the basic properties of sequences relate to similarity.

2 Second, the algorithm should yield a similarity-value that is relative to the

maximum potential similarity, resulting in a value between 0 and 1 (Abbott and Tsay, 2000; Elzinga, 2007).

3 Third, the algorithm should not undervalue order compared to length of sequence or to type of tokens (Dijkstra and Taris, 1995; Elzinga, 2003).

4 Fourth, the algorithm should be able to take into account theoretical knowledge about the data.

   4.1 Generally, one should be able to set to what extent certain events or statuses are similar to others. The common implementation is a substitution-matrix (or similarity-matrix) that can be filled through a multitude of methods (Gauthier et al., 2009; Hollister, 2009).

   4.2 Additionally, the algorithm should allow for different substitution matrices at different absolute time-positions (Aisenbrey and Fasaing, 2010; Lesnard, 2010).

   4.3 As discussed above, the algorithm could provide the flexibility to set the importance of certain similarity-types or specific similarities for the overall similarity-value of a sequence-pair (Abbott, 2000).

5 Fifth, it should be possible to adjust the importance of certain similarities or differences based on the immediate context of the involved elements (Hollister, 2009; Halpin, 2010).

6 Sixth, and compared to OM the most useful feature is the ability to, deal with multi-channel sequences, i.e. to deal with sequences that contain simultaneous elements (Abbott and Tsay, 2000; Gauthier et al., 2010).

It is ideally all of the above functionality to be included into any existing or new algorithm that calculates sequence similarities. As has been shown in the previous chapters, this is only partially the case for existing algorithm. Table 1 summarizes how current algorithms do or do not hold up against the desired functionality as presented above.

In short, OM based algorithms do not adhere to the Dijkstra and Taris principles because more order does not always lead to a higher similarity[11]. Additionally, they do undervalue order (Elzinga, 2003). With regard to time-dependent substitution matrices, these are only supported by Dynamic Hamming Matching and not by others (Lesnard, 2010). The setting of importancy for specific differences or similarities is not supported by any algorithm. Context is only taken into account by the algorithms that have specifically integrated this feature (Hollister, 2009; Halpin, 2010), in addition to partial support[12] by Elzinga (2003). Multi-channel is only partially supported by the

11 For example, ABCD versus ABDC (2 indels) has equal (dis)similarity as ACDB versus BCDA (2 indels). In the former, only the order of D-C has changed, while in the latter the order of A-B, A-C and A-D have all changed.

12 More on this in the next chapter

OM, Dynamic Hamming and MCSA method.

Table 1: overview of current algorithms and their functionality

Functionality	1	2	3	4.1	4.2	4.3	5	6
Existing algorithms	adhering to Dijkstra and Taris principles	yielding a relative distance	not undervalue order	supporting similiarity matrices	time-dependent substitution matrices	setting importance of differences	take context into account	supporting multi-channel sequences
Simple Optimal Matching (Abbott and Forrest, 1986)	no	yes	no	yes	no	no	no	no
Recombinational OM (Abbott and Tsay, 2000)	no	yes	no	yes	no	no	no	partial
DT coefficients (Dijkstra and Taris, 1995)	yes	yes	yes	yes	no	no	no	no
Dynamic Hamming Matching (Lesnard, 2010)	no	yes	no	yes	yes	no	no	partial
OMv (Halpin, 2010)	no	yes	no	yes	no	no	yes	no
Localized OM (Hollister, 2009)	no	yes	no	yes	no	no	yes	no
Elzinga Subsequence (Elzinga, 2003, 2007)	yes	yes	yes	no	no	no	partial	no
MCSA (Gauthier et al., 2010)	no	yes	no	yes	no	no	no	partial

It can be seen that no existing algorithm provides functionality across the full spectrum. The next chapter will illustrate how it is possible to create an algorithm that provides all functionality present in the scheme above.

4. Journey towards a better method :

Starting from a simple basic idea and developing it into a complex algorithm that provides more functionality than any current algorithm

Considering the desired functionality as presented in the previous chapter, and considering how the current algorithms fail on fulfilling these wishes, this chapter proposes a new method that provides all desired functionality.

The chapter builds from a simple idea towards the much more complex final result, the Mulder algorithm. For every step towards the Mulder algorithm, it is shown why the exact algorithm at that step is not yet providing all the desired functionality, and why consequently the algorithm needs to be taken still onwards onto the next step. This elaborate manner of presenting the Mulder algorithm serves two important purposes.

First, it enables the reader to understand the thoughts behind the creative process of building the Mulder algorithm. This not only deepens the understanding of the working and the goals of the algorithm itself, but additionally provides the possibility for researchers to create new algorithms by branching off at an intermediary step and building from there.

Second, it shows that the algorithm is not unnecessarily complex. Indeed, it is only the final Mulder algorithm that provides all the desired functionality, in contrast to any less complex previous iteration of the algorithm that only provides a subset of the desired functionality.

Base properties

The general nature of the new algorithm is based on the idea that any sequence can be described as a set of sociological characteristics, much like one can describe a person by describing the presence and position of head, arms, legs etc. The result is a distinct list of characteristics for every sequence, whereby it is very well possible that some characteristics (e.g. "doing activity A") occur in more than one sequence.

Multichannel sequences are described just like any single-channel sequence: one could for example generate characteristics like "doing A while doing B" or create both "A before B" and "C before B" when the sequence contains simultaneous activities A and C, followed by a single B.

Every characteristic has a value of importance coupled to it. Researchers can set this value to indicate that some specific characteristic is more important than another. The lowest value is 0, the default value is 1 and there is no maximum value. A value of 2 for a specific characteristic means that that characteristic is twice as important as a characteristic with the default value of 1.

To calculate the similarity between two sequences, it is simply checked for every characteristic of the sequences whether that characteristic occurs in both sequences. If a characteristic matches (i.e. occurs in both sequences), it is added to a list of matching characteristics. If all characteristics are checked, a value is calculated that is the sum of all importancy-values in the list of matching characteristics. The same value is calculated for the total combined list of characteristics in the two sequences, so that there are now two numbers: one representing the cumulative value of importance for the matching characteristics, and one representing the cumulative value of importance for all characteristics. The value that represents the similarity between the two sequences under study can be calculated by dividing the value for the matching characteristics by the value for the total list of characteristics. See figure 3 on the next page for an example.

figure 3 : an example to illustrate characteristics and their relation to similarity

Consider two sequences and their respective characteristics as below.

Sequence	Characteristics	Importancy value
AB	A before B	1
C	C before B	1
	A while C	2
AB	A before B	1

It can be seen that the "A before B" characteristic appears in both sequences, while the other characteristics don't. The sum of importancy values for the matching characteristics is 2 (value of 1 from the first sequence, and 1 from the second sequence), wihle the the total sum of importancy values for all characteristics is 5.
The total similarity-value between the sequences is the value of matching characteristics divided by the potential matching-value of all characteristics. In this case, that is 2 / 5 or in other words, the sequences are 40% similar.

It can be seen that characteristics with a high value of importance will impact the similarity-value more than characteristics that have a low value of importance. This can be illustrated by considering a characteristic with the lowest possible importancy-value of 0. Such a characteristic does evidently not contribute to the cumulative value of the matching list (i.e. adding 0 does not change the value) nor to the cumulative value of the overall list. Thus, when using these cumulative values to calculate the similarity between the two sequences, the impact of a characteristic with importancy-value 0 is absent. On the other hand, when using characteristics with ever higher importancy-values, the impact of those characteristics on the final similarity-value increases with every step.

Note that the similarity-values will always be a value between 0 and 1. This satisfies functionality 2 from the previous chapter. Additionally, the use of importancy-values is a first start for functionality 4.3 regarding the the ability to define some similarities (i.e. matching characteristics) as being more important than others, and the ability to analyse multi-channel sequences provides functionality 6. See table 2 for an overview.

Table 2 : overview of functionality of the base algorithm

Functionality Algorithm	1 adhering to dijkstra and taris principles	2 yielding a relative distance	3 not undervalue order	4.1 supporting simiilarity matrices	4.2 time-dependent substitution matrices	4.3 setting importance of differences	5 take context into account	6 supporting multi-channel sequences
Simple implementation	no	yes	no	no	no	partial	no	yes

Within the group of description-based methods, there are different choices with regard to the exact implementation. Very simple implementations of the method will be fast, but will not contain the functionality as described in a previous chapter. On the other end of the spectrum are the algorithms that do contain all the functionality but give in on speed.

Because it is the specific aim of this paper to include all functionality as previously described, the simple algorithms will only be touched upon very briefly while the algorithms with higher complexity will be described in more detail. The next chapter will be an in-depth discussion of a specific implementation of such a higher-complexity algorithm.

Simple method

The simplest form of the description-based algorithms is the one that generates very simple characteristics and also tests them in a very simple and unnuanced manner. Examples of generated characteristics are "doing activity A"[13] or the more detailed "doing activity A at time-position 10". The way in which characteristics are checked for occurrence in both sequences is a simple name-based boolean one: either the characteristic fully matches (i.e. the characteristic-name occurs in both sequences) or it doesn't at all. In terms of computational complexity, these algorithms are faster than those used in the traditional OM methods[14]. However, the name-based boolean nature of the checks gives problems for matching near identical characteristics. If one sequence had characteristic "doing activity A at time-position 10" and the other sequence had "doing activity A at time-position 9", there would be no match since they are not completely identical. This is not desired, since the only difference between the characteristics is a small change in timing. Table 3 provides an overview of the functionality for the simple implementation of a description-based algorithm.

Table 3: functionality overview of simple implementation of a description-based algorithm

Functionality	1	2	3	4.1	4.2	4.3	5	6
Algorithm	adhering to dijkstra and taris principles	yielding a relative distance	not undervalue order	supporting simiilarity matrices	time-dependent substitution matrices	setting importance of differences	take context into account	supporting multi-channel sequences
Simple implementation	no	yes	no	no	no	partial	no	yes

13 Note that only having characteristics like "doing activity X" will completely oversee the notion of order, which would make the algorithm barely useful.

14 The computational complexity of the simple algorithm is O(n1+n2) while the OM algorithms are O(n1*n2). The terms n1 and n2 stand for the amount of sequence-elements in respectively the first and second sequence of the pair. Note that the simple algorithm is O(n1+n2) because the amount of generated characteristics is linear to the amount of sequence elements and the algorithm only loops over all the characteristics once.

Tree-method

A way out of this problem is to create characteristics in a tree-like structure. The root characteristic would represent the full sequence (e.g. "ABC"), while child characteristics would represent ever smaller portions of the sequence (e.g. "AB", "BC" and ultimately "A"). If any large characteristic is not matched, chances are that some smaller subcharacteristic will still match. This solves the problem of not matching near identical characteristics, because even though the total characteristic is not completely matched, there is still similarity due to its matching child characteristics.

The subsequence algorithm of Elzinga (2003; 2008) is an example of such an algorithm. It creates characteristics by deleting any possible part of a sequence, and saving the remaining parts of the sequence as a characteristic. For example, from the sequence "ABC" one would obtain the characteristics "ABC", "AB", "BC", "AC", "A", "B" and "C".

Evidently, a tree-generating algorithms does not simply yield characteristics that describe isolated elements, but is more advanced than the simple algorithm in that it generates characteristics that describe an element in its context. The larger a characteristic, the more context it defines, and also, consequently, the more order it defines. This makes the tree-algorithms adhere to the Dijkstra and Taris principles about matching tokens and matching order.

Additionally, when only unique sub-characteristics are used (e.g. when the characteristic "A" is only generated once for a sequence "AAA", discarding the other two "A" characteristics), the algorithm will partially take context into account. This because adding an extra activity to a long spell (e.g. adding a B to the BBBB in BBBBC) will result in less extra characteristics than adding an extra activity in a short spell (e.g. adding a C to the C in BBBBC). Considering the fact that only the extra characteristics (i.e. the characteristics that the longer sequences has, but the shorter doesn't) will result in a mismatch between the two sequences, it can be concluded that adding an element to a long spell results in a relatively small distance, and adding an element to a short spell results in a relatively high distance. Thus, the impact of adding an extra element is partially context-dependent (Hollister, 2009; Halpin, 2010). See figure 5 for an example.

figure 5 : example to illustrate the partial support for context-dependency

Pair	Sequence 1	Sequence 2	Sociological difference
1 (Base sequence versus base + B)	BBBBC	BBBBBC	Small difference. The amount of extra unique characteristics in sequence 2 versus 1 is relatively low, and thus the difference is also low.
2 (Base sequence versus base + C)	BBBBC	BBBBCC	Higher difference. The amount of extra unique characteristics in sequence 2 versus 1 is relatively high, and thus the difference is also higher.

A downside to unique characteristics is that it will make the analysis of multi-channel sequences more difficult. If "A" happens 10 times at position 1 of "ABC", there will still be one unique "A", one unique "AB" and one unique "ABC" characteristic, the same result as when there would have been 20 or 100 or 2 "A" activities happening at time-position 1. Thus, it makes the calculation of similarity impossible for sequences where a certain activity can happen multiple times at the same time-position.

Thus, there is always a choice to be made between partially taking context into account (i.e. using unique characteristics) or fully supporting multi-channel (i.e. using all characteristics).

Regardless the choice, a downside to any tree-based method is that the use of substitution matrices is not really possible. One would have to switch from a simple boolean check (i.e. characteristics match completely or they do not at all), to some sort of calculation to check if maybe characteristic "AC" is approximately the same as "AD" when the similarity between C and D is 90%. However, "AC" might be even more similar to "DC" or "AF", depending on the substitution-matrix. Thus, the algorithm would need to compare "AC" with many other characteristics, finding a best match, and so for every characteristic. This might be possible on the practical side, but it gets difficult on the theoretical side when comparing longer characteristics like "ABCDC" with the "ABXDE". If C and X have 10% similarity, do the characteristics still match, and to what extent? And what is the match between "ABCDC" and "ABCDX" with the 10% similarity between C and X at the end? Should "ABCDC" pair with "ABXDE" or with "ABCDX"? These problems might have made Elzinga (2003; 2008) decide not include substitution matrices in his algorithms.

Table 4: functionality overview of the tree- implementation of a description-based algorithm

Functionality	1	2	3	4.1	4.2	4.3	5	6
Algorithm	adhering to dijkstra and taris principles	yielding a relative distance	not undervalue order	supporting similarity matrices	time-dependent substitution matrices	setting importance of differences	take context into account	supporting multi-channel sequences
Tree-implementation with all characteristics	yes	yes	yes	no	no	no	no	yes
Tree-implementation with unique characteristics	yes	yes	yes	no	no	partial	no	partial
Elzinga (2003)	yes	yes	yes	no	no	partial	no	no

Mulder method

Considering the usefulness of substitution matrices for SA to include theoretical knowledge about the data in an analysis, it would be of value to have a method as above, but one that could actually take substitution matrices into account.

Since it is the longer characteristics in the tree-algorithms that result in theoretical problems, the workaround is to only generate short characteristics, describing the relation between only two elements. More specifically, the characteristics describe the presence of two elements and detail their relation in terms of distance (i.e. how far are the elements apart) and direction (i.e. comes the second element before or after the first element). The name of a characteristic could be a simple summary of the characteristic. An example characteristic would be

"A_1_B" = { fist element : "A", second element : "B", distance between elements : 1 }

If A would not come before, but rather after B, the characteristic would become

"A_-1_B" = { fist element : "A", second element : "B", distance between elements : -1 }

where the distance between the elements is identical, but their order is reversed so that the distance turns from a 1 into a -1. A result of having the possibility to put negative distances is that any characteristic can be stored in two ways, with any of the two elements first or last. For example, the above "A_-1_B" can also be written as

"B_1_A" = { fist element : "B", second element : "A", distance between elements : 1 }

which describes the exact same relation between A and B. The algorithm also allows any element to have a relation with itself, so that sequences of length 1 (e.g. "C") will still end up having a characteristic.

Unique characteristics cannot be used since the lack of larger characteristics results in the same set of unique characteristics for two different sequences. It is outside the scope of this paper to extensively discuss this claim, but this is best illustrated by seeing that AAABBAABBB and AAABBBABBB will yield the exact same set of unique characteristics.

Contrary to the tree-method, the Mulder-method can work with substitution matrices. The substitution matrices can describe how much any element is different from an other element, so that for example the characteristic "A_1_B" becomes completely similar to "A_1_C" when the substitution-matrix describes B as being identical to C. Time-dependent substitution matrices can also be used when saving the absolute time-position of the first element[15], but it is obvious that setting time-dependent substitution matrices is an inherent difficult task, even more so than with normal substitution matrices. Indeed, when "A_1_B" and "A_1_C" happen at different absolute time-positions, how should one define the (dis)similarity between B and C? How much is having children in one era different from having no children in another era? These questions seem very hard to answer, but the method at least provides the functionality to test certain assumptions and help new research by supporting substitution matrices in its widest form.

15 Subsequently, the time-position of the second element can be calculated by using the distance between the two elements: the time-position of the second element is equal to the time-position of the first element plus the distance between the elements.

A result from supporting substitution matrices however, is that the algorithm cannot simply do a boolean check for the presence of a characteristic, but has to actually calculate a similarity between individual characteristics. More specifically, the algorithm has to compare every characteristic with every other characteristic in order to yield how much the characteristics are similar in terms of not only element-similarity (i.e. using a substitution-matrix) but also in terms of order and distance. Consequently, it is necessary to define how order, distance and element-similarity relate to distance-values. The algorithm should not definitely set this relation but rather let the researcher decide how he thinks that element-similarity, order and distance are important for the similarity between characteristics, and thus ultimately for the similarity between sequences. Combining this freedom with the already present allowance for setting importancy-values for individual (sets of) characteristics, it can be seen that the Mulder-method provides the full functionality 4.3 as hinted at by Abbott (2000).

When every characteristic has been compared with every other, the algorithm stores its results in a mirrored two-dimensional matrix. The rows stand for the characteristics of the first sequence while the columns of the matrix stand for the characteristics of the second sequence in the sequence-pair. Every cell consequently holds the similarity-value for a pair of characteristics from the different sequences. See the example in figure 6. Note that it is possible that multiple characteristics (e.g. "A_1_A" and "A_2_A" in the example) share the same characteristic ("A_1_A") in their highest matching pairs.

figure 6 : an imaginary partial similarity-matrix for unique characteristics. D and B are somewhat similar.

Seq1 Seq2	A_1_A	A_2_A	A_3_D	A_2_D	A_1_D	D_1_B
A_1_A	1	0.9	0	0	0	0
A_1_B	0	0	0.5	0.63	0.8	0
A_2_C	0	0	0	0	0	0

After a similarity-matrix for the characteristics has been created, the algorithm needs to determine an overall similarity-value for the sequence-pair. This leaves many possible implementations and thus makes it inherently difficult to choose for a specific one. There's at least the following possible implementations.

1. The "all best matches" method. Every characteristic could be given its best match, after which all the similarity-values of these matches are averaged, resulting in an overall similarity-value for the sequence-pair. Certain characteristics can be matched multiple times. For example, taking from the example in figure 6, the "A_1_A" and "A_2_A" from sequence 1 in would both

match with "A_1_A" resulting in a score of 1 and 0.9 respectively. A positive side to this method is that it is fast and simple. A downside is that many characteristics (e.g. the "A_1_A", "A_2_A" etc. from an AAAAAAAA sequence) can match one characteristic from the other sequence (e.g. "A_1_A" in AABCDEFGH), resulting in very high overall similarity-values.

2. The "multi-use penalty" method. The idea would be to give a penalty when a certain characteristic is matched multiple times. That way, a long sequence AAAAAAAAAA matching with sequence AA will receive penalties on the "A_1_A" characteristics because they match the one "A_1_A" from the AA sequence multiple times. This prevents the sequences from getting very high, and sociologically incorrect, similarity-values as with the "all best matches" implementation. Note that when some characteristics are penalized (e.g. from 1 to 0.5), it might only then become evident that they would have been better off pairing with a different characteristic in the first place (e.g. receiving 0.7 instead of the penalized 0.5). Thus, the algorithm needs to iteratively apply penalties, check whether there are characteristics that need to be re-paired, recalculate the scores and penalties in the new situation, and repeat the process until the overall similarity-value for the sequence-pair does not go up with the next iteration. Evidently, this is an intensive process which makes the implementation rather slow in some cases.

3. A method to prevent such an iterative process is to find a good estimate for the penalty: the "estimated penalty" method. One would need to find an approximation for the extent in which characteristics are used multiple times. We find such an estimate by considering two extreme cases. In the first case, all characteristics of sequence 1 would want to match with one specific characteristic of sequence 2 and not with other characteristics of sequence 2. In such a case, it can be seen that those latter (non-popular) characteristics of sequence 2, when asked with whom they would want to pair, would choose any of the characteristics of sequence 1, despite the fact that these characteristics of sequence 1 don't want to match with them. Thus, it is evident that when certain characteristics are used multiple times, this is reflected in the amount of unmirrored matches. An unmirrored match could be defined as a characteristic from one sequence that wants to pair with a characteristic from the other sequence while the latter wouldn't want the same.

   An additional illustration to the finding that unmirrored matches reflect the multiple-use of characteristics is the case of two identical sequences. When sequences are completely identical, every characteristic will pair with its exact equivalent characteristic in the other sequence and thus no characteristic is used multiple times.

It is seen that unmirrored matches represent multi-use of characteristics, and thus penalty, while less unmirrored matches represent rather single-use of characteristics and thus no penalty, or rather: full match.

The chapters on validation and algorithm-comparison will show how the different options compare and will detail on their specific use-cases, advantages and disadvantages. In any case, the Mulder algorithm supports most all desired functionality, except that context is not always taken into account. This is a small step back from the tree-algorithms, where context was baked into the longer characteristics. The Mulder algorithm however only uses short characteristics that describe two elements and the algorithm doesn't think in terms of 'insertion' or 'deletion' as to distinguish between an indel in a long spell or an indel in a short spell (see Hollister, 2009; Halpin, 2010). Still, the algorithm is sensitive for the distance between identical elements, as well as for the fact that adding a heterogeneous row of elements to a sequence is more substantial than adding a homogeneous spell to the same sequence. Functionality 5 with regard to context can thus be said to be 'partial'.

Table 5: functionality overview of the Mulder algorithm

Functionality	1	2	3	4.1	4.2	4.3	5	6
Algorithm	adhering to dijkstra and taris principles	yielding a relative distance	not undervalue order	supporting simiilarity matrices	time-dependent substitution matrices	setting importance of differences	take context into account	supporting multi-channel sequences
Mulder algorithm	yes	yes	yes	yes	yes	yes	partial	yes

The next chapter details the in-depth process of creating a similarity-matrix for the characteristics, and also discusses the subsequent calculation of an overall similarity-value by using the "multi-use penalty" implementation. The "all best matches" and "estimated penalty" implementation are discussed only briefly, because they are less complex and thus benefit less from an in-depth explanation.

5. Computing similarities using Mulder :

an in-depth explanation of the Mulder algorithm and the introduction of three different implementations of the algorithm

As seen in the previous chapter, it is possible to theoretically create an algorithm that calculates (dis)similarity between sequences while having most all currently desired functionality. This chapter details a specific implementation of such an algorithm and shows its internal workings in pseudo-code. Variables are prepended with a dollar sign while functions are highlighted in bold text.

First, three variables are defined.

The first variable is the Array `$sequences`, holding all sequences. It is not really important how these sequences are coded, as long as every sequence has a unique key and it is possible to generate a time-line like

```
channel 1            A A C C C C C C C
channel 2            B B B B   D D
time-position        1 2 3 4 5 6 7 8 9
```

This timeline is stored in a `$timelines` Array, where every Arrayelement has the same key as the sequence it was derived from. Every element contains an Array of time-positions, where every Arrayelement contains another array with the sequence-elements at that time-position. Taking the previous sequence as an example, this will result in

```
$timelines = Array(
      "seq1" => Array(
            "1" => Array("A"),
            "2" => Array("A", "B")
            ...
            "9" => Array("C")
      )
)
```

The second variable is `$substitutionMatrices`, holding different substitution matrices. The obligatory "default" substitution matrix defines the (dis)similarity between elements in general. Other substitution matrices can be defined (but are not required) for different absolute time-positions, as to enable for time-dependent definitions of (dis)similarity (Aisenbrey and Fasang, 2010; Lesnard, 2010). Every matrix is a simple two-dimensional Array that describes all possible combinations of the available element-types in terms of similarity with a value between 0 and 1. It is up to the researcher whether and how he fills the different substitution matrices.

The third variable is `$sequenceSimilarityMatrix` in which the end-result will be stored. It is another two-dimensional Array, but then crossing every sequence and in the end holding a similarity-value for every possible combination.

The general algorithm proceeds by first creating all characteristics for all sequences, and then calculating a similarity-value for every possible pair of sequences by comparing their respective characteristics. Pseudo-code is below. The total process will be detailed in the remainder of this chapter.

```
// Calculate characteristics and store them in $characteristicsPerSequence
$characteristicsPerSequence = Array();
foreach($timelines as $sequenceID => $timeline){
      foreach($timeline as $timeposition => $elements) {
            foreach($elements as $element) {
                  // Create characteristics for this element
                  $characteristicsPerSequence[$sequenceID] +=
                  createCharacteristicsForElement
                  ($element,$timeposition, $timelines);
            }
      }
}
// Calculate similarity-values for every pair of sequences
$sequenceSimilarityMatrix = Array();
foreach($sequences as $sequence1ID => $sequence1) {
      foreach($sequences as $sequence2ID => $sequence2) {
            // Cross the characteristics of $sequence1 with $sequence2
            // Result is a similarity-matrix for the characteristics
            $length = length($sequence1) + length($sequence2);
            $characteristicSimilarityMatrix =
            crossCharacteristicsOfSequences
            ($sequence1ID,$sequence2ID,$length);
            // Finally, go over the characteristic-matrix, and calculate
            // an overall similarity-value for the sequence-pair
            $sequenceSimilarityMatrix[$sequence1ID][$sequence2ID] =
            calculateSimilarityFromCharacteristics
            ($characteristicSimilarityMatrix);
      }
}
```

As can be seen, the algorithm starts by generating all characteristics for all the sequences. For every sequence, the algorithm starts at the very left of the sequence, iterates over all the elements present at that time-position, executes `createCharacteristicsForElement()` for every element, and proceeds to the adjacent time-position at the right. It repeats this process until the end of the sequence.

The `createCharacteristicsForElement($element,$timeposition)` generates the actual characteristics for the specified $element and adds them to the list of characteristics for the overarching sequence. The way in which the characteristics for an element are generated is through looping over all elements that are at either the same or a higher time-position compared to the `$timeposition` of `$element`, and describing the relation between that element and the current `$element`. The characteristics are the form of an Array holding the name and position of the alphabetically first element, and the name of and distance to the alphabetically second element[16]. Taking the previous sequence as an example, a characteristic of the first A element would be

```
$characteristic = Array(
      "firstelement" => "A",
      "firstelementposition" => 1,
      "distance" => 3,
      "secondelement" => "B",
      "importance" => [float]
)
```

The "importance" value is generated by running the $characteristic through a certain function `getImportancyForCharacteristic($characteristic)` and assigning the result to `$characteristic["importance"]`. How this function determines the importancy-values is completely up to the researcher or program that implements the function.

Note that the algorithm only describes relations with the elements to the right of any investigated element. Going to the left is always redundant, since they are already covered by previously generated characteristics. For example if `$element` was element B at time-position 4, it would be unnecessary to describe the relation with element A at time-position 1 (i.e. an element to the left) since this characteristic would have already been generated while describing that very first element A. The resulting characteristic would in both cases be the one in the example above.

Also note that if the algorithm was to describe the relation between a B at time-position 1 and an A at time-position 2, the "firstelement" would still be A, since it comes alphabetically before B. The value for the "distance" key would be "-1".

16 If there is no time-dependent substitution matrix available for the elementtypes and time-positions of the first or second element, it is theoretically not necessary to include the "firstelementposition", since this value will only be used to load the proper substitution matrix for elements at a certain time-position.

Once all characteristics are generated[17], and the algorithm loops over all the sequences, the **crossCharacteristicsOfSequences**`($sequence1ID,$sequence2ID,$length)` generates a matrix that holds how much every characteristic of the one sequence is similar to the characteristics of the second sequence. Its parameters are respectively the ID of the first sequence, the ID of the second sequence, and the cumulative length of the two sequences, the latter being used for normalization. The resulting matrix is used to match every characteristic of the one sequence with its most identical characteristic of the other sequence. This is easily understood when imagining two identical sequences that have the very same set of characteristics. The final similarity-value is obviously 1, but to arrive at this value it is necessary that every characteristic of the one sequence matches with its (completely identical) equivalent from the second sequence, and not with a (less similar) other characteristic. A sample matrix of a pair of identical sequences could be

```
$characteristicSimilarityMatrix = Array(
      1 => Array(1 => 1.0, 2 => 0.6, 3 => 0.8),
      2 => Array(1 => 0.6, 2 => 1.0, 3 => 0.7),
      3 => Array(1 => 0.8, 2 => 0.7, 3 => 1.0),
)
```

Where any nth characteristic of the first sequence has a perfect match with the nth characteristic of the second sequence, so that when any sequence is paired with its highest match, this would result in an average match of 1, i.e. two identical sequences, which is correct.

The similarity-value for a pair of characteristics (i.e. the similarity-value as found in any cell in the above matrix) is calculated by taking into account three variables:

1. `$a` represents the similarity between the elementtypes in the characteristics. Consider the comparison of two characteristics

   ```
   $1 = Array("firstelement => "A", "distance" => 2, "secondelement" => "C")
   $2 = Array("firstelement => "A", "distance" => -3, "secondelement" => "D")
   ```

   The algorithm then compares firstelement1 with firstelement2 using the `$substitutionMatrices["default"]` (or the specific time-dependent matrices, using the time-positions of the first and last element, if specified), which yields a certain similarity-value. The same comparison is executed for secondelement1 and secondelement2, yielding another similarity-value. Finally, $a is calculated as the product of the first and second similarity, resulting in a value from 0 to 1:

   ```
   $a = $similarity1 * $similarity2
   ```

17 If one opts for unique characteristics, every characteristic will occur only once. If one doesn't, some characteristics can occur mulitple times in the list of characteristics for a sequence. An example would be the sequence AAA with no time-dependent substitution matrices available. There would be two characteristics `Array("firstelement" => "A", "distance" => 1, "secondelement" => "A").`

2. `$b` represents the difference or equality of the signs of the distances. If the signs of the distances are identical (i.e. the distances are both >0 or both <0), `$b` will be 1. If the signs are different, `$b` will be 0. If one of the distances is 0, the sign of the other characteristic is irrelevant, so `$b` will be 1.
3. `$c` represents the absolute similarity in the "distance" value of the characteristics. A lower difference in distance results in a low `$c` and implies a high similarity.

   ```
   $c = abs($1["distance"] - $2["distance"])
   ```

   For a pair of characteristics, `$c` varies between 0 and abs(distance1) + abs(distance2). In a general sense for a pair of sequences, the maximum possible value of `$c` is consequently `the maximal possible distance between elements in the first sequence + the maximal possible distance between elements in the second sequence`. Thus, to normalize any `$c`, it is divided by the sequence maximum of `$c`, which equals (length(sequence1)-1)+(length(sequence2)-1) or in other words (`$length` – 2). Thus

   ```
   $c = ( abs($1["distance"] - $2["distance"]) ) / ($length -2)
   ```

   It can be seen that the resulting relative value now ranges between 0 and 1. To make the direction of `$c` in accordance with `$a` and `$b`, where a higher value means a higher similarity, `$c` is inverted. A higher `$c` now implies higher similarity.

   ```
   $c = 1 - $c
   ```

Now that `$a`, `$b` and `$c` are known, a total similarity-value for the pair of characteristics is calculated. This is done by running `$a`, `$b` and `$c` through a function that returns the similarity.

```
$similarity = similarityForABC($a, $b, $c);
```

The **`similarityForABC`** function to calculate the similarity for a pair of characteristics using the values of `$a`, `$b` and `$c` could for example be the following implementation

```
function similarityForABC($a, $b, $c)
{
      return $a*($b+$c)/2;
}
```

It can be seen that the total match ranges between 0 and 1. In this case, the match is 0 when either the elements of the characteristics don't match (i.e. `$a` is 0) or when the order (`$b`) and distance (`$c`) are 0.

Now that the code for calculating the similarity-value for a specific pair of characteristics is known, let us investigate the result for two characteristics where a substitution-matrix indicates that A and C are 100% similar.

```
$1 = Array("firstelement => "A", "distance" => 1, "secondelement" => "B")
$2 = Array("firstelement => "B", "distance" => -1, "secondelement" => "C")
```

It can be seen that the second characteristic also implies

```
$2a = Array("firstelement => "C", "distance" => 1, "secondelement" => "B")
```

which in turn is 100% identical to the first characteristic when taking into account the full similarity between the elements A and C. Still, if the algorithm was to calculate the similarity between `$1` and `$2`, it would yield a similarity of 0. Indeed, the first element A does not match with B, while the second element B does not match with C. Consequently, `$a` will be 0. Moreover, the order also yields a 0 for `$b`. The overall similarity value for the pair of characteristics will consequently be zero[18]. This is odd since it was just concluded that when characteristic `$2` was rewritten as `$2a`, it was easily seen that the similarity is 1 rather than the full opposite 0. Evidently, the algorithm needs to explicitly consider the alternative notations of one of the characteristics[19]. In other words, the similarity between characteristics (i.e. a cell in the `$characteristicSimilarityMatrix`) needs to be calculated as follows

```
$similarity1 = similarityBetweenCharacteristics($1, $2)
$similarity2 = similarityBetweenCharacteristics($1, inverseNotation($2))
$similarity = max($similarity1, $similarity2)
```

The resulting similarity is stored in the `$characteristicSimilarityMatrix`. Once all combinations of characteristics have been calculated, the matrix is full, and the function **`crossCharacteristicsOfSequences(`**`$sequence1ID,$sequence2ID,$length)` has ended. The next task is to calculate a similarity for the two sequences using the generated `$characteristicSimilarityMatrix` as input for the next function.

As has been discussed in the previous chapter, there are several options to calculate sequence-similarity from the characteristic similarity-matrix. The below paragraphs detail the implementation where a penalty is given to matches of characteristics that are used multiple times. The algorithm in general proceeds as follows

```
// Calculate the best matches for the characteristics of sequence 1
$seq1Matches = bestMatchesForCharacteristics
($characteristicSimilarityMatrix);
// Calculate the best matches for the characteristics of sequence 2
// Use a rotated similarity matrix with characteristic of seq 2 as keys
$rotatedSimilarityMatrix = rotate($characteristicSimilarityMatrix);
$seq2Matches = bestMatchesForCharacteristics
( $rotatedSimilarityMatrix );
```

18 This depends on the specific implementation of the **`similarityForABC()`** function, but in most implementations the total similarity will be 0 when both `$a` and `$b` are 0.

19 Only one characteristic needs to be rewritten. Rewriting the first is identical to rewriting the second, and rewriting both will yield a situation similar to the starting position of rewriting none.

```
// Generate statistics about how often the characteristics are used
$seq1CharacteristicUse = analyseUseOfCharacteristics
($seq1Matches, $seq2Matches);
$seq2CharacteristicUse = analyseUseOfCharacteristics
($seq2Matches, $seq1Matches);
// Calculate a score for the current set of matches
// Include a penalty based on how often certain characteristics are used
$similarityScore = calculatePenalizedScoreForMatches
($seq1Matches, $seq1CharacteristicUse, $seq2Matches, $seq2CharacteristicUse);

// Optimize matches until next optimalizations doesn't yield higher scores
$previousSimilarityScore = $similarityScore;
repeat
{
      $similarityScore = optimizeMatches
      ($seq1Matches, $seq1CharacteristicUse,
      $seq2Matches, $seq2CharacteristicUse,
      $characteristicSimilarityMatrix, $rotatedSimilarityMatrix);

      // Stop optimizing if scores only get lower
      if ($similarityScore < $previousSimilarityScore)
      {
            break();
      }
      else
      {
            // Store current similarityScore as previous score for next iteration
            $previousSimilarityScore = $similarityScore;
      }
}
```

The different functions that are used to come to the optimal solution will be described below. Most functions are rather uncomplicated, with the exception of the **`optimizeMatches`** one.

First , the **`bestMatchesForCharacteristics($similarityMatrix)`** function calculates the best matches for a set of characteristics by simply looping over the match-values of every row of the similarity-matrix, picking the highest and saving that as a match between the row-characteristic and the column-characteristic. Note that it finds the highest match for a specific *row* and thus finds the highest matches for the characteristics of sequence 1 when we use the `$characteristicSimilarityMatrix`.

Consequently, for finding the best matches for sequence 2, the algorithm uses a rotated characteristic-similarity-matrix `$rotatedSimilarityMatrix` that inverses the rows and columns: the rows are now the characteristics for sequence 2 and the columns are those for sequence 1.

Second, the **`analyseUseOfCharacteristics($matchSet1, $matchSet2)`** function analyses the use of certain characteristics in the matches. It stores for every characteristic of the first set the use of those characteristics in the matches of the second set. One element from the resulting Array could for example be

```
$seq1CharacteristicUse["A_1_A"] = Array(
      "count" => 3 // This is the total count. From seq1 > seq2 and vice versa.
      "wantsMatchWith" => Array("A_2_A"),
      "isWantedAsMatchFor" => Array("A_2_A", "A_3_A")
);
```

It can be seen that this particular characteristic "A_1_A" from sequence 1 is used 3 times. One time when the sequence 1 characteristic itself pairs with the "A_2_A" characteristic from sequence 2. And two times when the "A_2_A" and "A_3_A" characteristics from sequence 2 want to be paired with the sequence 1 characteristic. These numbers are used when calculating the sequence similarity-score for a certain set of matches. Note that when comparing two identical sequences, all characteristic-matches are mirrored so that `"wantsMatchWith"` and `"isWantedAsMatchFor"` contain the very same list and `"count"` is always 2.

Thirdly, calculating the penalized similarity-score using the aforementioned analysis and the `$seq1matches` and `$seq2matches` is done using the **`calculatePenalizedScoreForMatches`** function. The first step of the function is to iteratore over the `$seqCharacteristicUse` Arrays and check the usecount of every characteristic. If the use-count is more than 2 (i.e. more than with a mirrored match), the algorithm will assign a penalty factor to the match-values of all matches that use that specific characteristic. More specifically, for a certain characteristic

```
$penaltyFactor = 0.5;
for all $match that uses $characteristic
{
      $score = $score * $penaltyFactor;
}
```

Thus, when certain characteristics are used more than twice, the matches in which the characteristic appears, will be halved[20]. When all penalties have been applied, the function runs over all the matches and returns the average match-value as the

20 This value seems to work best. One could also use elaborate formulas like `$penaltyFactor = 1 - min( 0.9, ($characteristic["count"]-2) * 0.05 )` but these perform worse than simple values >= 0.5.

similarity-score for the pair of sequences that the characteristics and matches were derived from.

Evidently however, once penalties are applied, some characteristics would have had a higher match when they had been paired with a different characteristic (e.g. one that does not end up receiving a penalty). This is what the **`optimizeMatches`** function is used for. It loops over the characteristics that have received a penalty, and tries to lower the "`count`" by assigning a "`isWantedAsMatchFor`" characteristic to a different characteristic. It then executes the **`calculatePenalizedScoreForMatches`** function on the new set of matches and analyses how the similarity-score changes. After it has tried all re-assignments of characteristics, it returns the best score it could get, and saves the sets with which this results was obtained. As can be seen in the overall algorithm two pages back, the returned result is compared with the existing similarity-score and if the new score is higher, the **`optimizeMatches`** function is executed again, in the hopes for an even better score when re-assigning even more characteristics. Once the scores don't improve anymore, the **`optimizeMatches`** function is not called again, and the resulting similarity-score for the pair of sequences is stored.

Results show that the optimizing of scores is a necessary step, sometimes improving similarity-scores with more than 25%. Still, as has been discussed, the optimization process is quite intensive in terms of computing power. In the worst case, every re-assignment of characteristics has effect, so that the algorithm needs to continue optimizing, resulting in re-assigning every characteristic. For every such step, the algorithm loops over every characteristic trying out a re-assignment, executing the **`calculatePenalizedScoreForMatches`** function for every new configuration in the process. The result is an O(n*n*n) function. On the positive side, the scores often quickly stop improving, and only few characteristics[21] have a "`count`" higher than 2, so the actual complexity will more likely be between O(n*n) and O(log(n)*n), but it is still useful to also investigate the simpler solutions as presented in the previous chapter.

The next chapter will discuss the possible implementations of the Mulder algorithm in terms of how they perform sociologically in arbitrary tests, while the chapter thereafter will compare the best-performing Mulder-implementation with the existing algorithms on usefulness with several data-sets.

21 When not using unique characteristics, a certain characteristic (e.g. "A_1_A") can sometimes pair with one of many identical characteristics (e.g. 5 "A_1_A" characteristics, all scoring 1). It is then advised to assign the match to the characteristic that has the lowest use-count. This prevents situations where one characteristic gets penalized for having many matches, while other identical characteristics have far fewer matches.

6. Validation of the Mulder algorithm :

whether the Mulder algorithm yields sociologically meaningful results, and which implementation of the algorithm is best in doing so

This chapter will both validate and benchmark the three different implementations of the Mulder algorithm and investigate their performance. Validation is concerned with the absolute results of the three implementations of the Mulder algorithm, while benchmarking is concerned with their relative performance.

Validation ensures that the Mulder algorithm functions theoretically and sociologically correct before the next chapter investigates how the algorithm performs better than other algorithms. Tests are executed in two different ways.

First, small data-sets are used to see whether a set of changes to a base sequence yields corresponding similarity values in the expected direction[22]. Evidently, there are no formal rules about how certain changes should lead to a certain outcome, and thus the expected directions for any change are rather based on literature as well as on triangulated results from different actors. The extent to which the results of the different implementations correlate with the expected outcome, determines their sociological performance, and it is Hollister (2009, pp. 250) that describes this "correlation with subjective ratings" as a "strong candidate" for benchmarking OM-like algorithms. Note that due to limited space, only the absolute results of the tests are included. This implies that the tests in this chapter only investigate the *direction* of results (i.e. is the outcome expected or not) but not the *magnitude*. Thus, even when the different implementations of the algorithm would all yield expected absolute directions, it can very well be that one implementation performs sociologically better (i.e. more in line with what people would expect) than the other implementations in its subtleties. The last summarizing paragraph will also take these subtleties of results into account, and for reference, the full results with all the calculated distances between the sequences can be found in Appendix 1 : Validation.

The second type of benchmarking investigates the relative speed of the different implementations for a larger data-set. Indeed, one algorithm could very well be so slow that it becomes unusable, despite having a sociological advantage.

A final paragraph of the chapter discusses the different outcomes both in terms of validation and in terms of benchmarking. The implementation of the Mulder algorithm that performs best in terms of sociologically expected results and speed will be used in the next chapter to be benchmarked against existing algorithms, as used in literature.

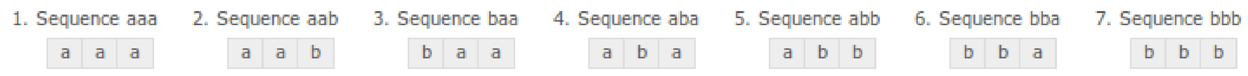
22 Note that these are minimal tests. Correctly passing the tests does not guarantee that the tested functionality works as desired or expected in all cases: results should be seen as a guidance.

Testing sociological direction of results

Test 1 : Dijkstra and Taris principles

The first test concerns whether the algorithms formally adhere to the Dijkstra and Taris principles (Dijkstra and Taris, 1995; Elzinga, 2003). Consider the sequences in sequence-set 1.

sequence-set 1 : sequences for the first validation test



The tested principles are

1. Sequences that have no elements in common are maximally dissimilar. Thus, the similarity between sequence "aaa" and "bbb" should be 0.
2. Sequences that have the same spells in the same order are maximally similar. Thus, every sequences matching with itself should be yielding a similarity of 1.
3. The more elements that sequences have in common, the more similar these sequences are. Thus, taking "aaa" as a base, the similarity to "aab", "aba" or "baa" should be higher than to "abb" or "bba". Additionally, the similarity to "abb" or "bba" should be higher than to "bbb".
4. The more common order there is among common elements, the more similar sequences are. Thus, sequence "aab" should be more similar to "abb" than to "bba". Even so, sequence "baa" should be more similar to "bba" than to "abb".

The results for the different implementations in table 6.

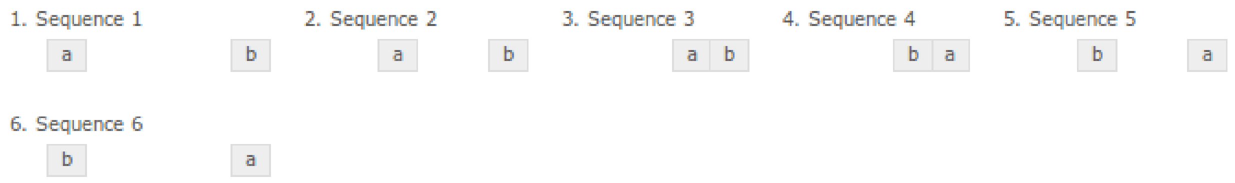
table 6 : results for the first validation test

Mulder-implementation	Principle 1	Principle 2	Principle 3	Princple 4
1. All best matches	Pass	Pass	Pass	Pass
2. Multi-use penalty	Pass	Pass	Pass	Pass
3. Estimated penalty	Pass	Pass	Pass	Pass

Test 2 : Order and distance

The second test determines how the different implementations handle order and distance. Consider the single-channel sequences in sequence-set 2.

sequence-set 2 : first set of sequences for the second validation test



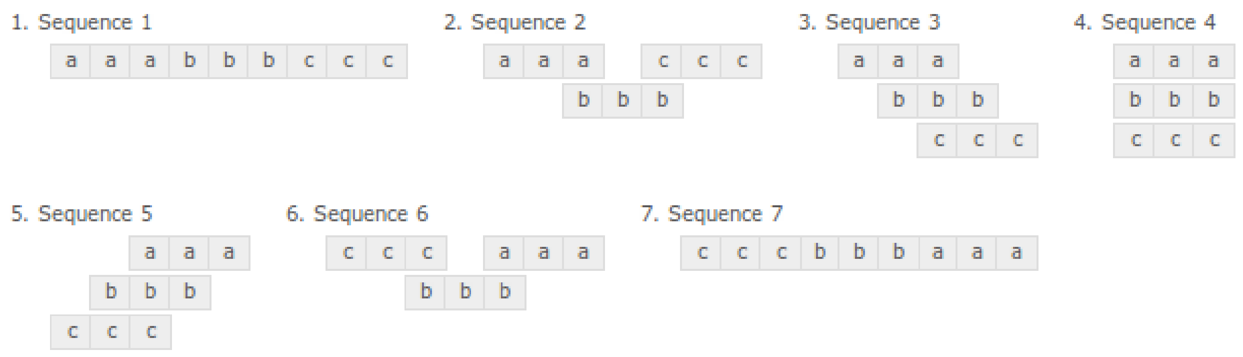
It can be seen that in the set of sequences, the "b" and "c" move from respectively the very left and very right to the opposite position in the sequence, thereby reversing the order of the initial sequence 1 into the opposite sequence 6. Consequently, starting from sequence 1 as a base and comparing it with the other sequences, the similarity to the subsequent sequences is expected to decrease, being highest for sequence 2 and lowest for sequence 6. The results can be found in table 7.

table 7 : first results for the second validation test

Mulder-implementation	1-2 > 1-3	1-3 > 1-4	1-4 > 1-5	1-5 > 1-6
1. All best matches	Pass	Pass	Pass	Pass
2. Multi-use penalty	Pass	Pass	Pass	Pass
3. Estimated penalty	Pass	Pass	Pass	Pass

If the simple single-element spells from the previous sequences are replaced by longer spells, the sequences turn into multi-channel when the different spells overlap. To illustrate this behaviour, an additional 'c' spell is added. The resulting set of sequences can be found in sequence-set 3.

sequence-set 3 : second set of sequences for the second validation test



It can be seen that the blocks of "aaa" and "ccc" move from their initial positions at the left and right to the exact opposite position. In the process, the order of the "aaa", "bbb" and "ccc" blocks gets reversed. Again, when starting from sequence 1 as a base and comparing it with the other sequences, the similarity to the subsequent sequences should decrease, being highest for sequence 2 and lowest for sequence 7. The results can be seen in table 8.

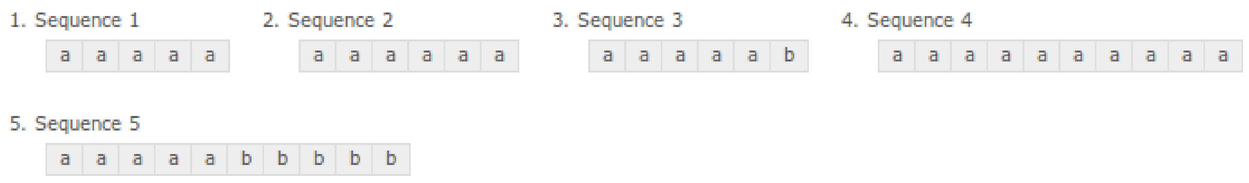
table 8 : second results for the second validation test

Mulder-implementation	1-2 > 1-3	1-3 > 1-4	1-4 > 1-5	1-5 > 1-6
1. All best matches	Pass	Pass	Pass	Pass
2. Multi-use penalty	Pass	Pass	Pass	Pass
3. Estimated penalty	Pass	Pass	Pass	Pass

Test 3 : addition of elements

Consider the sequences in sequence-set 4.

sequence-set 4: first set of sequences for the third validation test



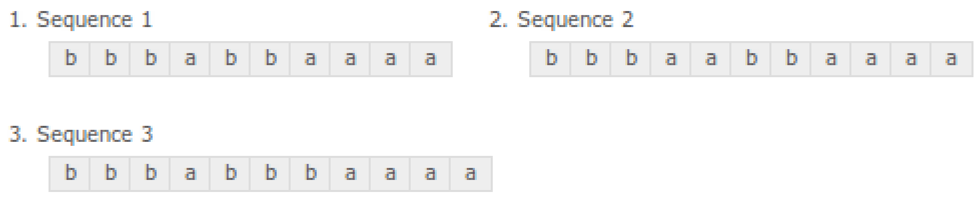
Taking sequence 1 as a base, the other sequences all add specific elements to that base. Sequence 2 and 4 increase the length of the "aaaaa" spell, while sequence 3 and 5 add a spell of "b" elements. Sociologically, a change in timing is less substantial than adding a completely new activity (source), and thus sequence 1 should be more similar to sequence 2 than to sequence 3. Even so, the similarity to sequence 4 should be higher than to sequence 5. In addition, it can be seen that the change in sequence 4 is a more substantial version of the change in sequence 2, while the change in sequence 5 is a more substantial version of the change in sequence 3. Thus, the similarity to sequence 4 should be lower than to sequence 2 and the similarity to sequence 5 should be lower than to sequence 3. The results can be seen in table 9.

table 9 : first results for the third validation test

Mulder-implementation	1-2 > 1-3	1-4 > 1-5	1-4 < 1-2	1-5 < 1-3
1. All best matches	Pass	Pass	Pass	Pass
2. Multi-use penalty	Pass	Pass	Pass	Pass
3. Estimated penalty	Pass	Pass	Pass	Pass

While the above results show that the addition of uncommon elements is more substantial than the addition of common elements, it is important to see whether this behaviour does not only hold at the scale of complete sequences, but also for smaller parts of sequences, so that the addition of an element to a long spell is less significant than the addition of an element to a short spell. Consider the sequences in sequence-set 5.

sequence-set 5: second set of sequences for the third validation test



Sequence 1 contains as many 'a' elements as 'b' elements, so the commodity of 'a' and 'b' is equal on the sequence level. In the middle of the sequence, it can be seen that there is a spell consisting of one 'a' element, as well as a spell consisting of two 'b' elements. The addition of an 'a' to the short spell should be more significant than adding a 'b' to the longer spell. See table 10 for results.

table 10 : second results for the third validation test

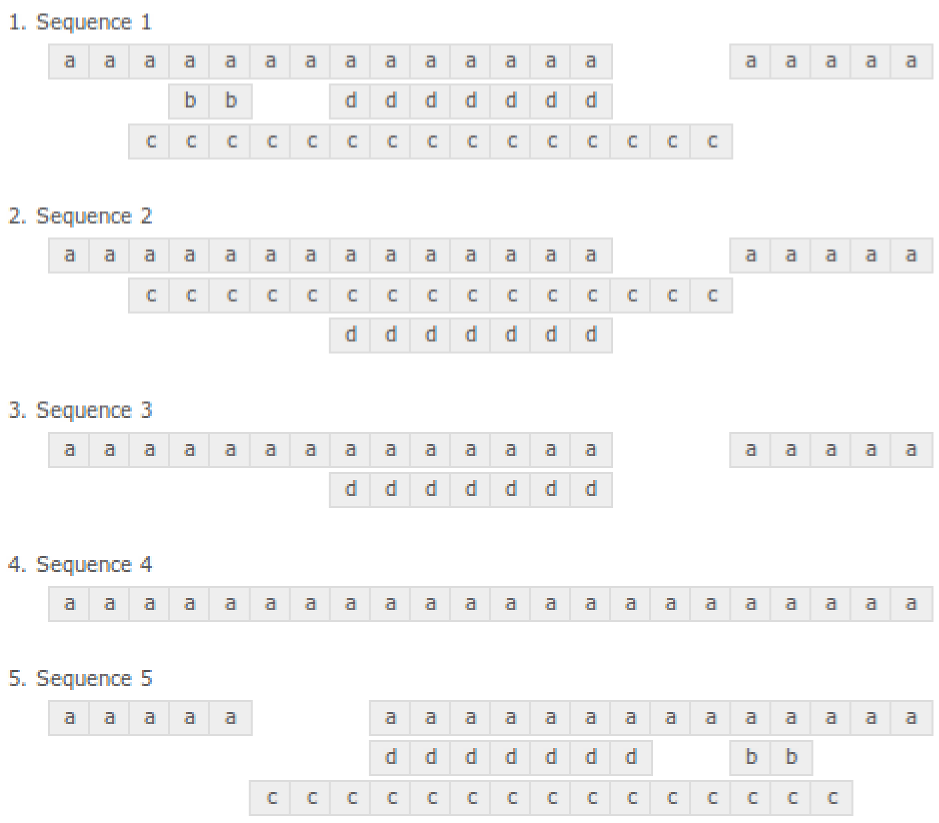
Mulder-implementation	Additional 'b' is more similar than additional 'a'
1. All best matches	Fail
2. Multi-use penalty	Pass
3. Estimated penalty	Pass

The simple 'All best matches' fails with this test. It finds that adding an 'a' or a 'b' is equivalent. In this case, it is thus better to use the more complex variants of the Mulder algorithm.

Test 4 : complex multi-channel sequences

This test is meant to illustrate and test how the different implementations of the Mulder algorithm handle complex multi-channel sequences. Consider the sequences in sequence-set 6, where sequence 1 and sequence 5 are each others reverse.

sequence-set 6: sequences for the fourth validation test


1. Sequence 1
a a a a a a a a a a a a a a a a a a a
b b d d d d d d d
c c c c c c c c c c c c c c c
2. Sequence 2
a a a a a a a a a a a a a a a a a a a
c c c c c c c c c c c c c c c
d d d d d d d
3. Sequence 3
a a a a a a a a a a a a a a a a a a a
d d d d d d d
4. Sequence 4
a a a a a a a a a a a a a a a a a a a a a a
5. Sequence 5
a a a a a a a a a a a a a a a a a a a
d d d d d d d b b
c c c c c c c c c c c c c c c

Taking sequence 1 as a base, and comparing it with the sequences 2 to 4, it is evident that the similarity goes down for every next sequence: pair 1-2 has the highest similarity, and pair 1-4 has the lowest. Additionally, if the algorithm is fair, it does not make any difference between the pair 4-1 and pair 4-5. Indeed, sequence 4 is identical from left to right or from right to left, so that sequence 1 and its reverse sequence 5 should yield an identical distance. The results can be found in table 11.

table 11 : results for the fourth validation test

Mulder-implementation	1-2 > 1-3	1-3 > 1-4	4-1 == 4-5
1. All best matches	Pass	Pass	Pass
2. Multi-use penalty	Pass	Pass	Pass
3. Estimated penalty	Pass	Pass	Pass

It can be seen that all variants of the Mulder algorithm pass the test.

Testing speed

The last test is concerned with the relative speed of the different implementations of the Mulder algorithm. Several datasets with multiple complex multi-channel sequences are analysed while the time it takes to do so is being recorded. The averaged and normalized results can be seen in table 12.

table 12 : results for the fourth validation test

Mulder-implementation	Time (normalized) to execute a set of comparisons
1. All best matches	1
2. Multi-use penalty	1.35
3. Estimated penalty	1.25

It can be seen that the simple 'all best matches' implementation is the fastest implementation. The other implementations take around 30% more time, whereby the 'estimated penalty' implementation is about 10% faster than the iterative 'multi-use penalty' method.

Note that this test does not include absolute results because these depend on both on the speed of the specific programming language the algorithm is implemented in, and on the computing power of the machine that the tests are executed upon. When the algorithm gets implemented in existing software package, this also opens up possibilities to compare the Mulder algorithm in terms of speed with existing algorithms.

Overall

It can be seen that the different implementations of the Mulder algorithm pass all the validation tests, except for the 'all best matches' method in test 3. Thus, the 'multi-use penalty' and the 'estimated-penalty' implementations can be considered valid in terms of sociological expectations, while the 'all best matches' implementation still performs well, but is off on how it handles the addition of elements in shorter or longer spells.

Relatively speaking, the 'all best matches' performs worst among the three variants of the Mulder algorithm, as seen in the above tests. A deeper investigation of the full results[23] confirms these findings. Additionally, it is found that the 'estimated-penalty' method performs best in terms of yielding sociologically expected results. This is surprising, since the 'estimated-penalty' method was originally meant as a simpler, possibly less accurate, version of the complex 'multi-use penalty' method, but now seems to outperform the other variants.

In terms of speed, the 'estimated-penalty' method is 25% slower than the fastest 'all best matches' method. However, it is 10% faster than the 'multi-use penalty' implementation. With regard to the latter implementation, it should be noted that the speed of the algorithm can drop dramatically in worst-case scenarios, due to its higher computational complexity (O(nˆ3) compared to the O(nˆ2) of the other implementations).

Considering the relatively low differences in speed between the implementations[24], and taking into account the better sociological performance of the 'estimated-penalty' implementation, it can be seen that the 'estimated penalty' seems to be the most promising variant of the Mulder algorithm.

Now, although the Mulder algorithm in the form of the 'estimated-penalty' implementation did pass all the validation tests, it is important to see how it performs when compared with existing algorithms. Only then can it be concluded whether the algorithm is actually better than current algorithms, in addition to being theoretically sound and yielding sociologically expected results. The next chapter provides comparisons with the OM algorithm (Abbott and Forrest, 1986; Pollock, 2007), the Hamming distance algorithm (source) and the recent Multi-Channel Sequence Analysis (MCSA) as proposed by Gauthier et al. (2010).

23 See the similarity matrices in appendix 1.

24 One could argue that speed is very important and that an 25% increase is significant, but especially with the ever increasing computational power of hardware, speed becomes less of an issue. Indeed, discussions in literature do revolve around sociological issues of the algorithms, and not around their relative speed.

7. Comparison with existing algorithms

and how the Mulder algorithm yields more useful results

The previous chapter has shown that the Mulder algorithm, and specifically the "estimated-penalty" implementation, yields sociologically expected results, as to validate the Mulder algorithm on a basic level.

It is out of the scope of this paper to illustrate how the Mulder algorithm compares to the existing algorithms on all possible features, but coming back to the research question, it is especially important that the results from the Mulder algorithm are better in the key parts of this research: multi-channel analysis, proper valuation of order, and time-dependent substitution matrices.

Comparisons are executed by analysing the results that the different algorithms produce for certain data-sets. While benchmarking the performance on a particular issue (e.g. multi-channel analysis), not all existing algorithms are compared, but rather a certain subset of the available algorithms. For example, when dealing with multi-channel sequences, it is evident that comparisons are only possible for the algorithms that actually support multi-channel sequences. For every comparison, a short introduction explains which algorithms will be compared. In general, the Mulder algorithm is compared with OM (Abbott and Forrest, 1986), OM with recombined tokens (Abbott and Tsay, 2000; Pollock, 2007), Elzinga subsequences (Elzinga, 2003; 2008), the Dynamic Hamming Distance algorithm (Lesnard, 2010) and the recent Multi-Channel Sequence Analysis (MCSA) as proposed by Gauthier et al. (2010).

Results for the OM and Dynamic Hamming algorithms are calculated through the R statistical package (Gabadinho et al., 2009; Gabadinho et al., 2011). The MCSA and Elzinga results are calculated through dedicated software for these methods.[25] The chapter proceeds as follows.

First, small tests show how the Mulder algorithm yields more useful results for specific input, more specifically multi-channel data and time-dependent substitution matrices.

Second, the results of the Mulder algorithm are compared to other algorithms with regard to the valuation of order.

Third, to demonstrate the usefulness of the Mulder algorithm for analysing real-life data, an existing dataset is used to investigate whether the calculated similarity-values are sociologically valid on an aggregate level using clustering techniques.

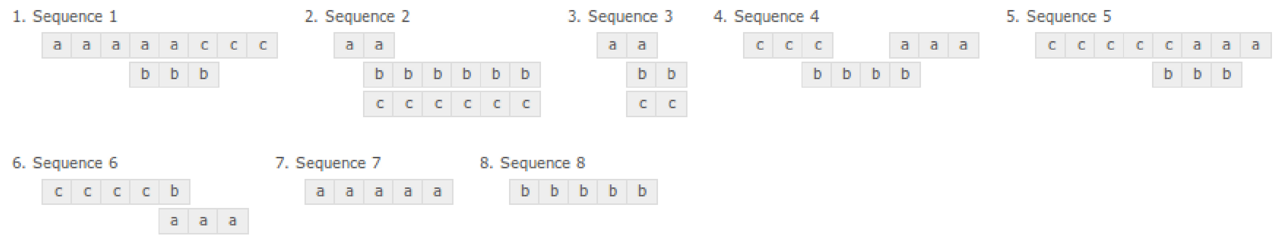
25 The Elzinga results are calculated through the CHESA package. See http://home.fsw.vu.nl/ch.elzinga/ .
The MCSA results are calculated through SALTT in T-coffee. See http://tcf_dev.vital-it.ch/apps/tcoffee/index.html

Specific input

Chapter 4 has argued on theoretical and grounds that the Mulder algorithm can yield sociologically valid results when using multi-channel sequences or time-dependent substitution matrices. The following test aim to test these theoretical assumptions.

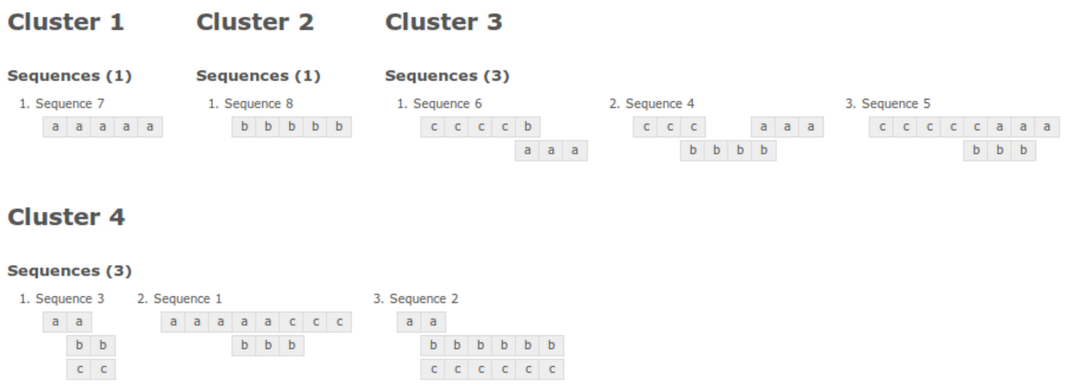
To compare the usefulness of the different algorithms with regard to multi-channel data, the following multi-channel algorithms are compared: OM (Abbott and Forrest, 1986), OM with recombined tokens (Abbott and Tsay, 2000; Pollock, 2007), Elzinga subsequences (Elzinga, 2003; 2008), the Dynamic Hamming Distance algorithm (Lesnard, 2010) and MCSA (Gauthier et al., 2010). Other algorithms cannot deal with the multi-channel nature of the data and can consequently not produce any results. Consider the sequences of sequence-set 7.

sequence-set 7: sequences for the comparison on specific input



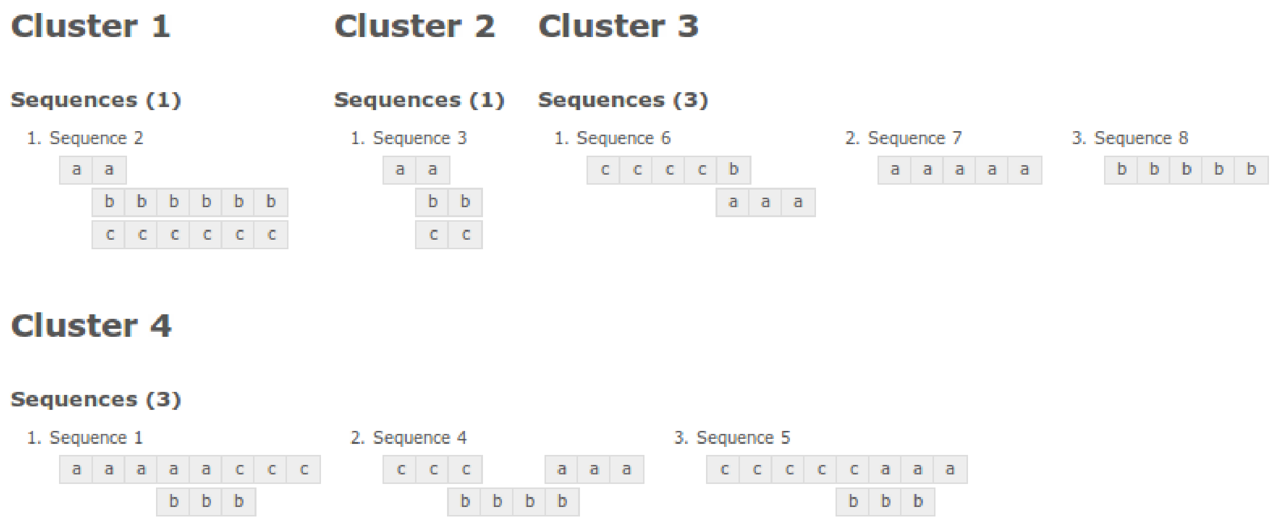
It can be seen that the data-set mixes both single-channel and multi-channel sequences, and plays with order. The first three sequences are in the same general order (a before b before c) while sequences 4 until 6 have a reversed order (c before b before a). The sequences 7 and 8 both miss the multi-channel properties of the other sequences and additional only exist of one element-type, whereas the other sequences all have 'a', 'b' and 'c' included. Sociologically, one would expect that the first three sequences are relatively similar (with the third sequence being most deviant), the next three sequences are relatively similar, and sequence 7 and 8 are rather on themselves. To test this assumption, it is calculated how certain sequences are close to each other for the different algorithms. The resulting clusters are shown below. The full distance matrices can be found found in Appendix 2. First, consider the outcome of the Mulder algorithm in cluster outcome 1.

cluster outcome 1 : cluster outcome for the Mulder algorithm while testing specific input



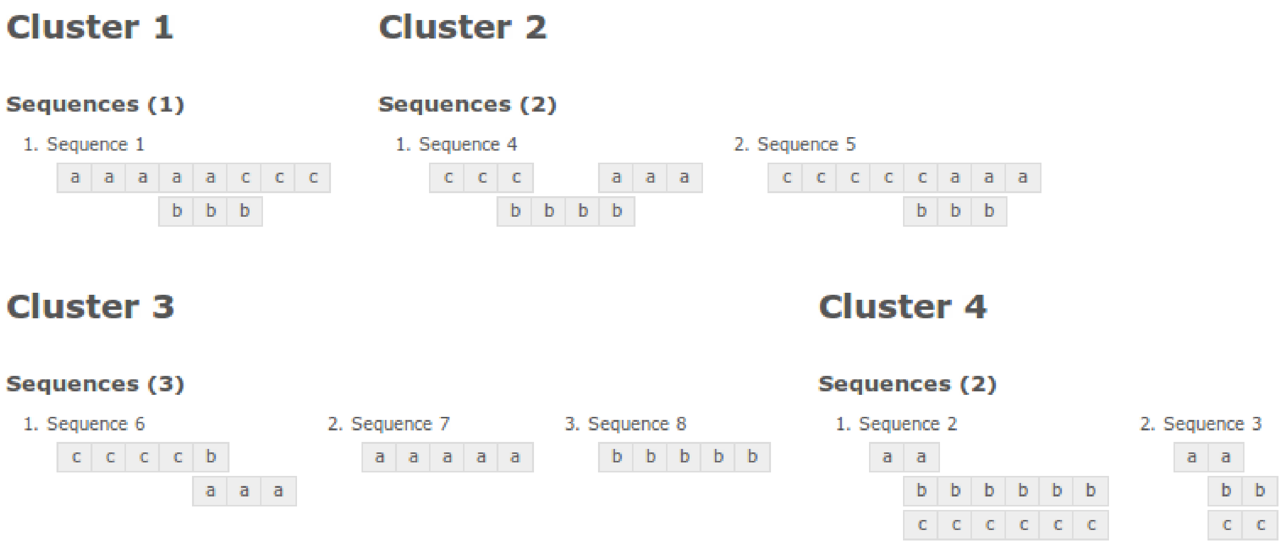
It can be seen that the groups are specifically as expected. Sequences 7 and 8 form their own cluster, while the other sequences are grouped together on order. As a comparison, consider the results for the OM algorithm in cluster outcome 2.

cluster outcome 2 : cluster outcome for the OM algorithm while testing specific input



Evidently, these results are unexpected and cannot be sociologically explained. First, the completely dissimilar sequences 7 and 8 grouped together in cluster 3, while they should evidently be in different groups. Second, cluster 4 contains sequences that have the exact opposite order. The reason might be that the OM algorithm is primarily concerned with the length of sequences, and thus groups sequences of equal length together. The tests on valuation of order will shed light on this issue. Next, consider the Dynamic Hamming results in cluster outcome 3.

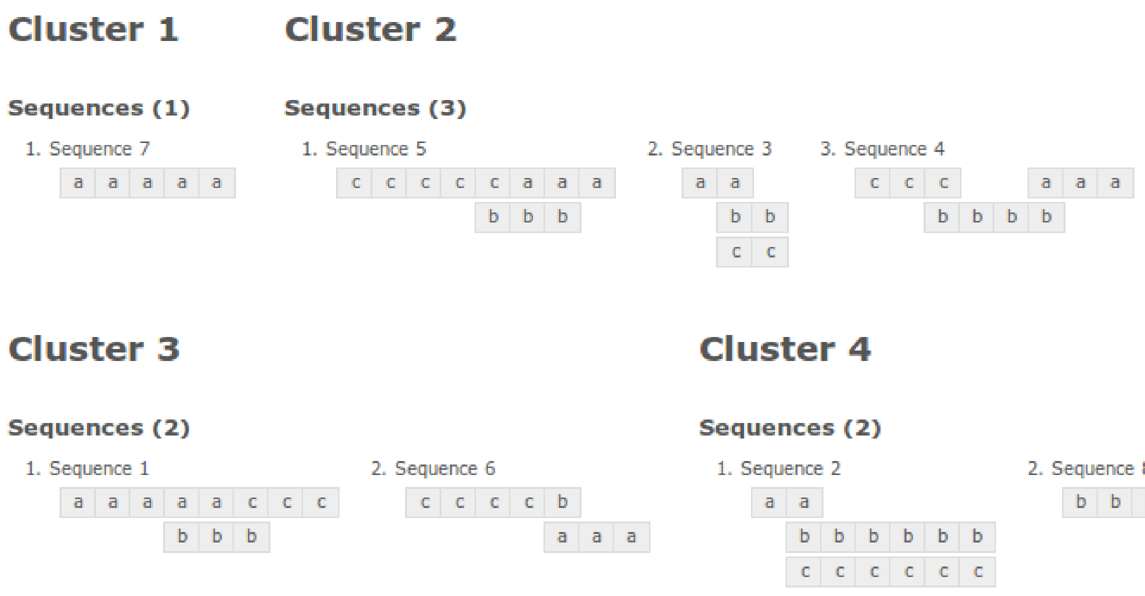
cluster outcome 3 : cluster outcome for the Dynamic Hamming algorithm while testing specific input



Again, results are unexpected. The dissimilar sequences 7 and 8 are still together in one cluster, and sequence 1 has a cluster of its own instead of being grouped with the sequences of identical order.

Now consider the outcome for the MCSA algorithm in cluster outcome 4.

Cluster outcome 4 : cluster outcome for the MCSA algorithm while testing specific input



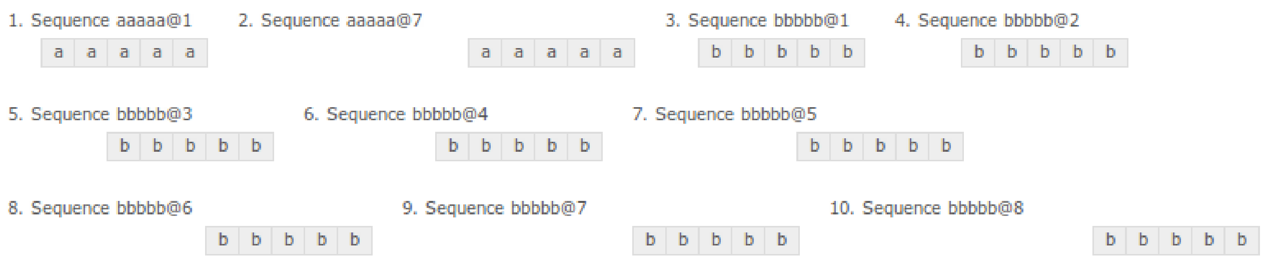
Still, results do not improve sociologically seen. First, the sequences of different order 1 and 6 are placed together in cluster 3. Second, sequence 8 does not have its own cluster but is rather grouped with sequence 2 that is quite dissimilar from it. Third and last, cluster 2 contains sequence 3 which might be a better fit for sequence 2 than the current sequences 4 and 5.

All in all, it is shown that all existing algorithms have problems with yielding sociologically expected results for the above set of multi-channel sequences. Other tests[26] indicate that existing algorithms perform especially bad on sets of sequences that have an unequal length and an unequal width (i.e. a different amount of channels for different sequences). For sequences with equal length and an equal number of channels, results are not so bad. This can be seen in the tests with real life data.

Next, the usefulness of different algorithms is tested while using time-dependent substitution matrices. The only existing algorithm that supports this is the Dynamic Hamming Distance (Lesnard, 2010) that was created specifically for this purpose. A limitation of the Hamming-type of algorithms is that it can only deal with sequences of uneven length, thus the following data-set only contains sequences of length 5.

sequence-set 8: sequences for the comparison on time-dependent sequences

26 Due to limited space not in this paper.



The time-dependent substitution matrices have been entered such that there are two era's, the first era from time-position 1 until 6 where 'a' is completely similar to 'b' and the second era from time-position 7 until 12 where 'a' is completely dissimilar to 'b'. Additionally, it is arbitrarily decided that sequence elements that cross borders of era's are completely dissimilar[27]. It can be argued that sequence 1 and 3 in era 1 should be identical while sequence 2 and 10 from era 2 should be completely dissimilar. Moreover, it can be seen that, taking sequence 1 as a base, sequence 3 until 10 should yield ever lower similarity values. Indeed, the 'bbbbb' sequence moves out of era 1, where 'a' and 'b' are similar, into era 2 where 'a' and 'b' are completely dissimilar.

The results can be seen in table 13. Maximum similarity is 1, minimum is 0.

table 13 : results for the implementation of time-dependent substitution matrices

	Similarity 1-3	Similarity 2-10	Similarity 1-3 until 1-10
Mulder algorithm	1	0	Decreasing from 1 to 0
Dynamic Hamming	1	0.5	Alternating

It can be seen that the Mulder algorithm produces the expected results. The Dynamic Hamming method however yields a value of 0.5 for the similarity between sequences 2 and 10 while the substitution matrix at that time indicates that the 'a' and 'b' elements are completely dissimilar. Apart from this obvious sociologically invalid result, the Dynamic Hamming also yields sociologically unexpected results on the second point. The Mulder algorithm correctly yields increasingly lower similarity values for the sequences 3 until 10, but the Dynamic Hamming method yields alternating results.

Clearly, the Mulder algorithm provides results that come closest to the sociologically expectations as postulated above, making the Mulder algorithm the most useful option for SA analysis when using time-dependent substitution matrices. Note that this is even more evident when using data of unequal length: indeed, as noted above, the Hamming-type algorithms are unable to deal with sequences of unequal length, making the Mulder algorithm the only viable option in that case.

27 The rationale for setting time-dependent substitution matrices is outside the scope of this paper; it is most important that the values in the time-dependent substitution matrices are taken into account in the way that one would sociologically expect.

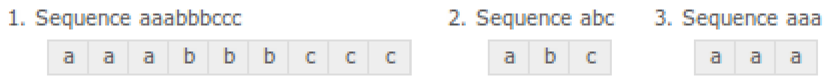
Valuation of order

The valuation of order is an important aspect for all algorithms. Indeed, order is an important sociological property of any sequence (Elzinga, 2003), such that any algorithm that undervalues order is less useful to yield sociologically valid (dis)similarity values.

Due to the importance of order and its evident presence in most any sequence, all major algorithms are included in the comparison. The Mulder algorithm is compared with OM (Abbott and Forrest, 1986), OM with recombined tokens (Abbott and Tsay, 2000; Pollock, 2007), Elzinga subsequences (Elzinga, 2003; 2008) in both the 'common subsequences' and the 'matching subsequences' implementations[28], and MCSA (Gauthier et al., 2010). The Dynamic Hamming Distance algorithm (Lesnard, 2010) cannot be used, since the Hamming-type of algorithms does not support sequences of unequal length.

The first test concerns single-channel sequences, so that all algorithm can take part in the comparison. Consider the sequences in sequence-set 9.

sequence-set 9: first set of sequences for the comparison on valuation of order



It is seen that sequence 1 and sequence 2 have identical order, only a difference in timing. Sequence 1 and sequence 3 have the 'a' elements in common, but the overall sequences are clearly different. The difference in length from sequence 1 to respectively 2 and 3 are identical. Thus, when comparing pair 1-2 with pair 1-3, it would sociologically make sense that the 1-2 pair is more similar than the 1-3 pair.

The resulting distance matrices for all the aforementioned algorithms are as following. Because the matrices report on dissimilarity rather than similarity, the cells that represent the distances between sequence 1 and 2 should have a *lower* distance value than the cells that represent the distance between sequence 1 and 3.

OM

	aaabbbccc	abc	aaa
aaabbbccc	0	1.33	1.33
abc	1.33	0	0.44
aaa	1.33	0.44	0

28 These two implementations differ on whether they only take the amount of unique subsequences into account ('common subsequences') or also consider the amount of occurences of a specific subsequence ('matching subsequences'). A more detailed discussion regarding the Elzinga measures can be found in Elzinga (2007).

MCSA

	aaabbbccc	abc	aaa
aaabbbccc	0	3400	0
abc	3400	0	6700
aaa	0	6700	0

Mulder

	aaabbbccc	abc	aaa
aaabbbccc	0	0.77	0.85
abc	0.77	0	0.87
aaa	0.85	0.87	0

Elzinga normalized matching subsequences

	aaabbbccc	abc	aaa
aaabbbccc	0	0.78	0.98
abc	0.78	0	0.88
aaa	0.98	0.88	0

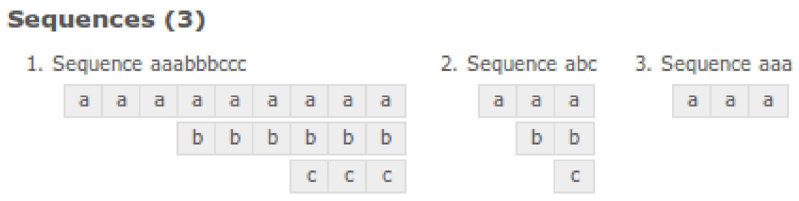
Elzinga normalized common subsequences

	aaabbbccc	abc	aaa
aaabbbccc	0	0.82	0.88
abc	0.82	0	0.82
aaa	0.88	0.82	0

It can be seen that only the Mulder and Elzinga algorithm yield sociologically expected results. It is also exactly those algorithms that have specifically been targeting order as an important aspect of input data. Note the especially deviating results of the MCSA (Gauthier et al., 2010) algorithm, suggesting that the 'aaabbbccc' and 'aaa' sequences are completely identical while obviously they are different. Also note that the OM algorithm regards 'aaa' and 'abc' as very similar and also as equally distant from 'aaabbbccc'. This is to be attributed to OM focussing on length, so that sequences of equal length are regarded as similar, and sequences of different length as dissimilar[29].

The next analysis turns to the role of order in multi-channel sequences. This is especially relevant, since there is currently no algorithm that supports multi-channel and also explicitly takes order into account. Indeed, the Elzinga (2003; 2008) algorithm does explicitly consider order, but has never been able to analyse multi-channel sequences. Consider the sequences in sequence-set 10.

sequence-set 10: second set of sequences for the comparison on valuation of order



The sequences are kept rather identical to the sequences in the previous single-channel analysis, but the sequences are now multi-channel. It can be seen that sequence 1 and 2 again have a generally identical order while the third sequence is more dissimilar from the other two sequences. The results for the different algorithms are as following. Note that there are no Elzinga results, since the Elzinga algorithm

29 This has been confirmed by extra tests. Due to space limitations, these tests are not shown in the paper.

does not support multi-channel sequences.

OM

	aaabbbccc	abc	aaa
aaabbbccc	0	3.78	3.33
abc	3.78	0	0.67
aaa	3.33	0.67	0

MCSA

	aaabbbccc	abc	aaa
aaabbbccc	0	3400	6700
abc	3400	0	6700
aaa	6700	6700	0

Mulder

	aaabbbccc	abc	aaa
aaabbbccc	0	0.76	0.96
abc	0.76	0	0.73
aaa	0.96	0.73	0

Again, it can be seen that the OM method does not consider order to be important: the shorter sequence 2 and sequence 3 are considered to be very similar, while sequence 1 and sequence 2 are considered to be very dissimilar. The Mulder algorithm does not dispute that sequence 2 and sequence 3 are quite similar, but it also fulfils the sociological expectation that sequence 1 and sequence 3 are quite similar as well, at least more similar than sequence 1 and sequence 2. Note again the sociologically unexpected behaviour of the MCSA algorithm: it finds that 'aaa' is equally similar to the relatively small sequence 2 as it is to the much larger and more complex sequence 1.

Real-life data : the family status and employment database

The previous comparisons have been using specific data-sets to test the usefulness of the different algorithms for multi-channel sequences, the use of time-dependent substitution matrices and the valuation of order. This comparison tests the usefulness of the different algorithms for analyses with real-life data. Not only is this comparison more objective since it uses existing data that has been used in other papers, it also provides insights in how the Mulder algorithm provides sociologically valid results on an aggregate scale, i.e. when analysing larger amounts of sequences.

The used database comes from the German Life History Study, a retrospective life course study that follows several cohorts of people over their life courses[30]. The data used in this comparison represents 233 West German women born in 1971. There are two channels. The first channel represents the employment status in the 20 months after the woman has first found a job. The second channel represents the family status in that same period of 20 months. The data-set only contains sequences that have

30 More information can be found on http://www.yale.edu/ciqle/GLHS/index.html

both channels present. Some sequences contain gaps. The different codes within the family and employment channels are respectively

A = single
B = married, no child
C = in a relationship
D = cohabiting, no child
E = married, child
F = cohabiting, child

G = 1st employer
H = side job
I = unemployment
J = 2nd employer
K = unpaid caregiving
L = 3rd employer

To ensure that the results do not result from luck, two independent random samples are drawn from the overall pool of sequences. For every sample, the following tests are executed independently and then compared. It is found that conclusions drawn from the tests are identical for the two samples. Thus, it can be said that the results did not derive from luck, but are grounded in the data. The two complete samples, along with full results are available for further research[31], but due to limited space it is not possible to show them completely in this paper. Instead, specific highlights illustrate the more general conclusions.

Indeed, conclusions are merely general and not sequence-specific. Contrary to the previous tests, it is less obvious whether specific similarity-values are sociologically correct, and it is additionally not feasible to analyse all the values individually. Thus, three aggregate indicators are used to determine the ability of the different algorithms for yielding sociologically expected results with larger data-sets.

The first indicator is concerned with analysing the average or 'representative' sequences, as distilled from the similarity-values that an algorithm produces. Representative sequences are those sequences that exhibit the key features of all sequences. They can be found by picking exactly those sequences that have a high average similarity to other sequences in the dataset. Once found, it can be analysed whether these calculated representative sequences are the sequences that one would sociologically expect to be representative for the dataset.

The second indicator is concerned with how the different algorithms produce distinct clusters that correlate with sociological expectations (Hollister, 2009). Indeed, if an algorithm groups sequences together in the exact same way that humans would group them together, then the algorithm is doing a good job at producing sociologically valid results.

31 The full clustering results can be found at http://fabricasapiens.nl/mulderalgorithm/

The third indicator builds upon the results from the second indicator. If the second indicator shows that the clusters from a certain algorithm make sociologically sense, the third indicator investigates how determined that algorithm is with its choice. This can be done by analysing the density of clusters, i.e. the extent to which the algorithm groups sociologically similar sequences closely together. An objective measure exist for this purpose.

First indicator : representative sequences

The analysis of representative sequences yields insights in how well the different algorithms are able to give high average similarity-values to exactly those sequences that have the most common sociological properties. Two different methods for finding representative sequences are used[32].

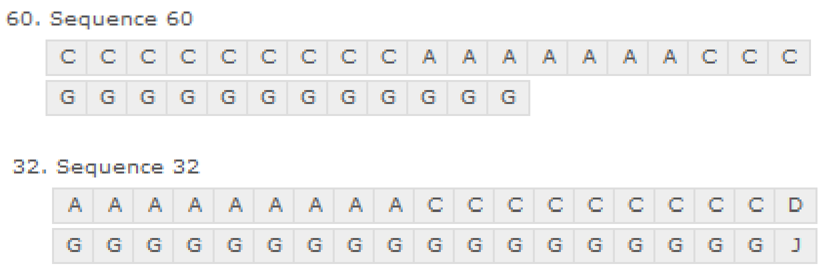
First, sequences are found for their 'centrality' in the overall data-set. Thus, if one envisions two 'clusters' of sequences (i.e. groups of sequences that have high inter-sequence similarity) and one sequence in between these groups, it would be that very sequence in the middle that would be the 'central representative sequence'.

A second method does not merely try to find the most central sequence, but rather tries to find the most central sequence in dense areas within the data-set. Thus, continuing with the previous example, this would find the central sequences of the two clusters, but leave the sequence in the middle untouched.

Analysis of the different generated representative sequences shows that representative sequences are mostly identical for the existing algorithms. Thus, existing algorithms seem to place the same sequences in the 'middle' of the dataset (centrality) and in the middle of the dense areas.

An interesting point of the Mulder algorithm is its central sequence, when compared to those of the existing algorithms. See the most central sequences for the Mulder and other algorithms in sequence-set 11. The first sequence is the central sequence of the Mulder-algorithm, the second sequence is the most central sequence in the other algorithms.

sequence-set 11: central sequence of the Mulder and other algorithms.



60. Sequence 60

C	C	C	C	C	C	C	C	C	A	A	A	A	A	A	A	C	C	C
G	G	G	G	G	G	G	G	G	G	G	G							

32. Sequence 32

A	A	A	A	A	A	A	A	A	C	C	C	C	C	C	C	C	C	D
G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	J

32 The representative sequences were found using the R statistical package, and more specifically the seqrep function. See Gabadinho (2009, 2011) for more details on seqrep and R in general.

While the existing algorithms have representative sequences that merely represent how the majority of sequences has certain elements in a certain place, it is the representative sequence of the Mulder algorithm that does implicitly include not only common elements, but also the common order in which they appear. It can be seen that 'being single' (A) is often followed by 'in a relationship' (C) and vice-versa.

The fact that the Mulder algorithms was able to catch this commonly encountered sociological phenomena, while the other algorithms could not, is a plus for the Mulder algorithm in terms of its ability to yield sociologically correct results.

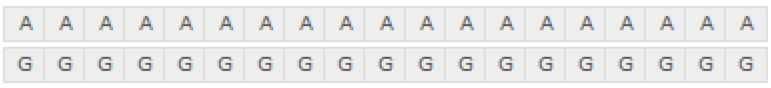
Second indicator : distinct clusters

A very strong indicator for the sociological performance of the discusses algorithms is how well one is able to create distinct clusters from the similarity-values they create (Hollister, 2009). Indeed, sociologically distinct clusters show that the algorithm was able to create high pairwise similarity-values for sequences that share a certain sociological aspect. Thus, the clustering results for the similarity-values of the different algorithms are investigated and conclusions are drawn for the ability of that specific algorithm to produce sociologically valid results.

After analysis[33], several observations are made.

First, it is seen that the larger clusters are identical for all algorithms, including the Mulder algorithm. The two largest clusters consist out of mostly identical sequences. The first large cluster consists of sequences representing single women, having a job at their first employer. See sequence-set 12 for an example of these sequences.

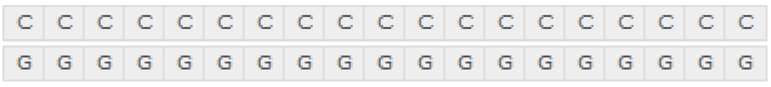
sequence-set 12: sequence of first big cluster



A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G

The other large cluster consists of sequences representing women in a relationship, also having a job at their first employer. See sequence-set 13 for an example of these sequences.

sequence-set 13: sequence of first big cluster



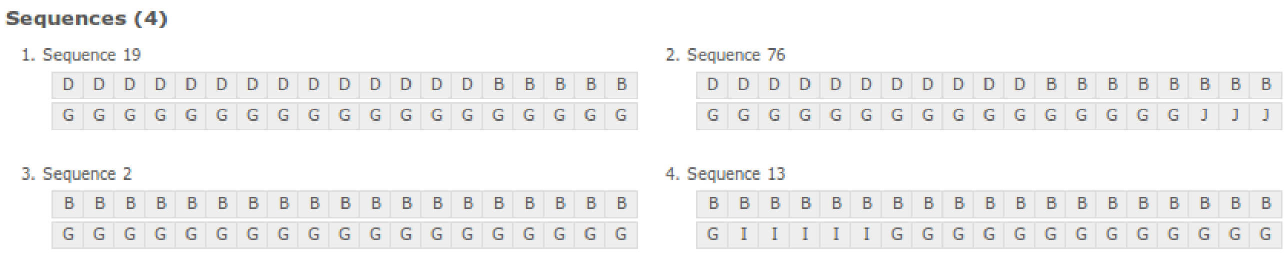
C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C
G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G

Concerning the smaller clusters, it can be seen that there are small deviations in how sequences are grouped. Some of these deviations are sociologically unfavourable. For example, the similarity-values of the OM and MCSA algorithm make the clustering

33 Algorithms are analysed on the amounts of clusters that are 'optimal', i.e. the amounts where clusters are most distinct. This amount does not differ significantly between algorithms.

algorithm sometimes place specific sequences in sociologically unexpected groups, i.e. the sequences would have sociologically better fitted somewhere else. An example would be sequence 76 and sequence 19 from sample 1 that with OM and MCSA are clustered with sequences with only 'B' elements in the first channel. See cluster-outcome 5.

Cluster outcome 5 : cluster outcome for the MCSA and OM algorithm for sequence 76 and 19



Sequences (4)

1. Sequence 19

D	D	D	D	D	D	D	D	D	D	D	D	D	D	B	B	B	B	B
G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G

2. Sequence 76

D	D	D	D	D	D	D	D	D	D	D	B	B	B	B	B	B	B	B
G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	J	J	J

3. Sequence 2

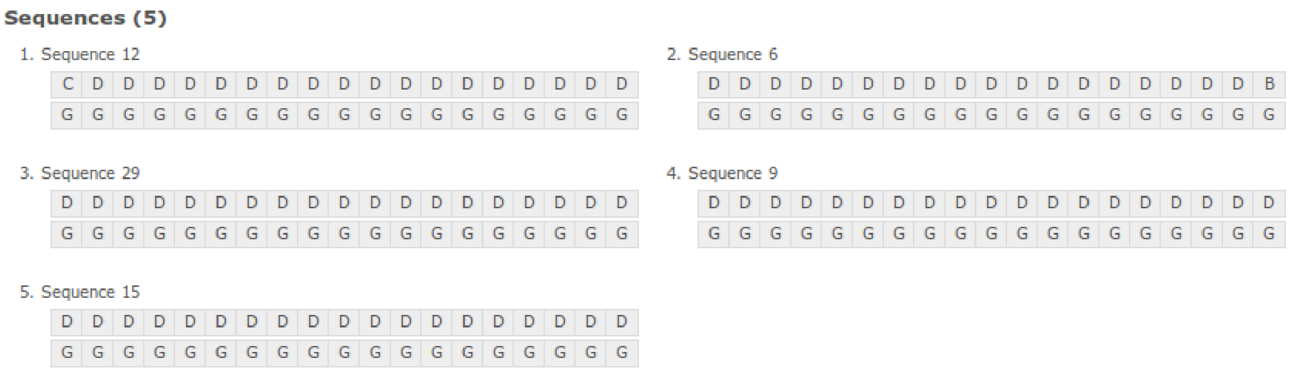
B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G

4. Sequence 13

B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
G	I	I	I	I	I	G	G	G	G	G	G	G	G	G	G	G	G	G

Sociologically, it would make more sense when these two sequences were clustered with sequences that have mainly 'D' elements in the first channel. Indeed, sequence 76 and 19 have many more 'D' elements than 'B' elements. For MCSA, it is sociologically more valid to group sequences 76 and 19 within one of the other available clusters, specifically the cluster in cluster outcome 6, as can be seen below. Note that all sequences in the cluster have only 'D' elements in the first channel.

Cluster outcome 6 : cluster where sequence 76 and 19 would fit better



Sequences (5)

1. Sequence 12

C	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G

2. Sequence 6

D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	B
G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G

3. Sequence 29

D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G

4. Sequence 9

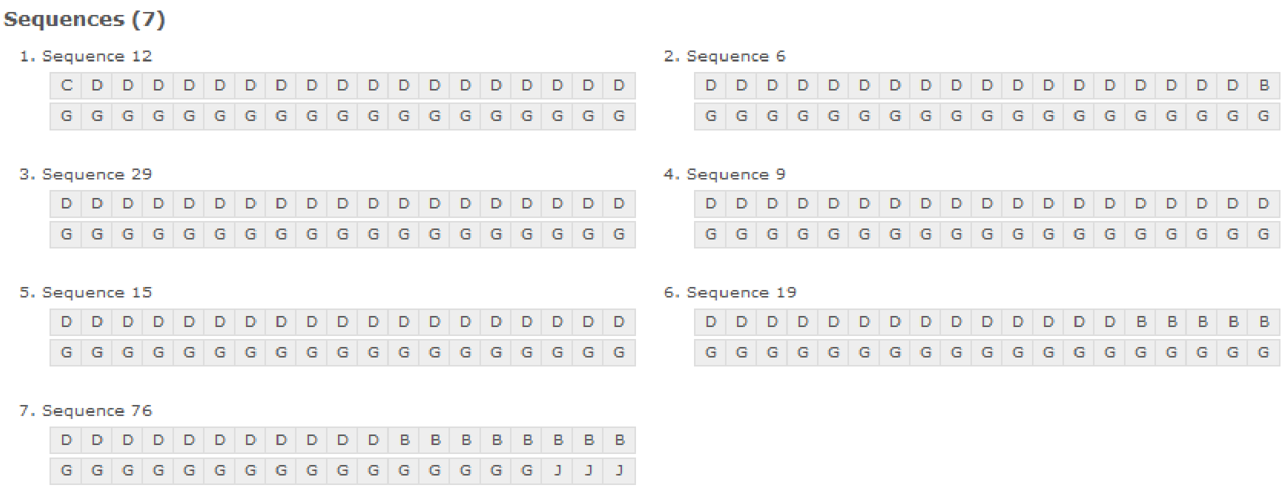
D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G

5. Sequence 15

D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D
G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G

Contrary to the unexpected results from the OM and MCSA method, the Mulder algorithm groups sequences 76 and 19 in a cluster with 'D' activities in the first channel. This is a sociologically more expectedly approach, as can be seen in cluster outcome 7 on the next page.

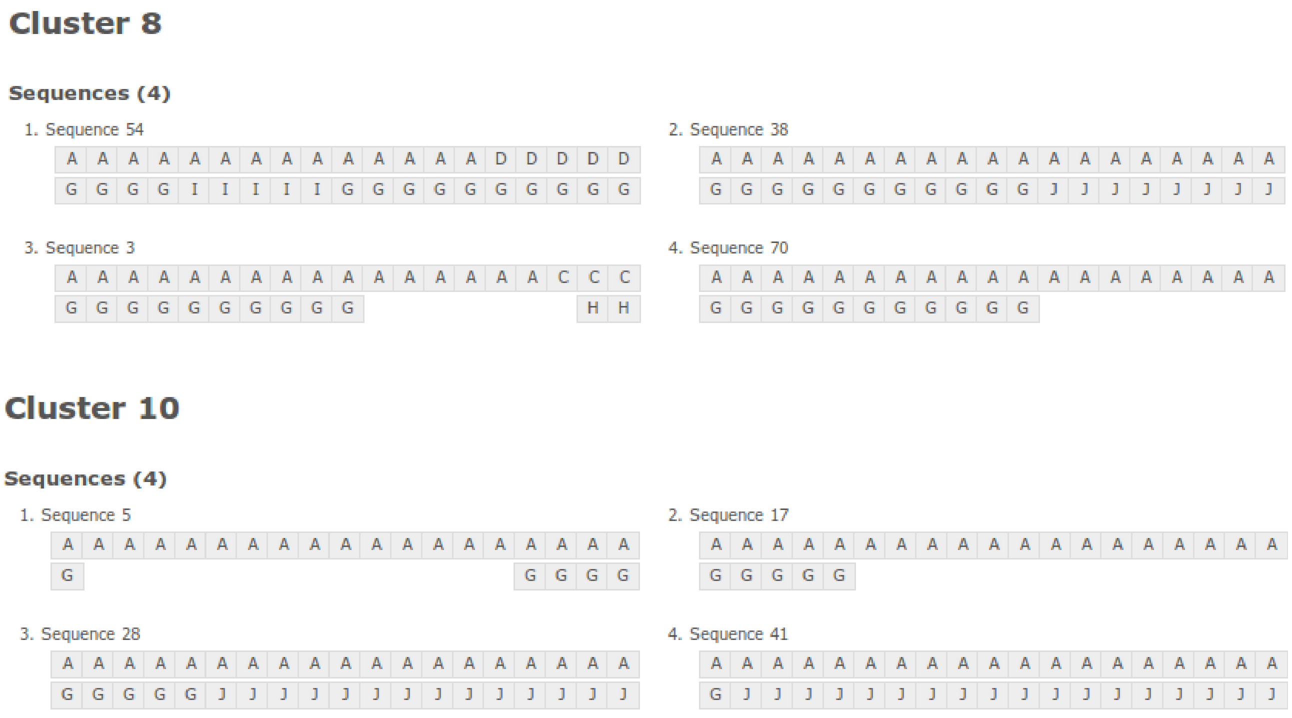
Cluster outcome 7 : cluster outcome for the Mulder algorithm for sequence 76 and 19



```
Sequences (7)

1. Sequence 12
   C D D D D D D D D D D D D D D D D D D
   G G G G G G G G G G G G G G G G G G G

2. Sequence 6
   D D D D D D D D D D D D D D D D D D B
   G G G G G G G G G G G G G G G G G G G

3. Sequence 29
   D D D D D D D D D D D D D D D D D D D
   G G G G G G G G G G G G G G G G G G G

4. Sequence 9
   D D D D D D D D D D D D D D D D D D D
   G G G G G G G G G G G G G G G G G G G

5. Sequence 15
   D D D D D D D D D D D D D D D D D D D
   G G G G G G G G G G G G G G G G G G G

6. Sequence 19
   D D D D D D D D D D D D D D B B B B B
   G G G G G G G G G G G G G G G G G G G

7. Sequence 76
   D D D D D D D D D D D B B B B B B B B
   G G G G G G G G G G G G G G G G J J J
```

With other tests, similar situations occur. Investigating OM and MCSA results, it becomes especially evident that the MCSA algorithm yields sociologically unexpected results. Analysis shows that often it is easy to point out how clusters could have been re-arranged to yield sociologically more valid results. An example would be the cluster in cluster outcome 8.

Cluster outcome 8 : certain cluster outcome for the MCSA algorithm



```
Cluster 8

Sequences (4)

1. Sequence 54
   A A A A A A A A A A A A A A D D D D D
   G G G G I I I I I G G G G G G G G G G

2. Sequence 38
   A A A A A A A A A A A A A A A A A A A
   G G G G G G G G G G G J J J J J J J J

3. Sequence 3
   A A A A A A A A A A A A A A A A C C C
   G G G G G G G G G G               H H

4. Sequence 70
   A A A A A A A A A A A A A A A A A A A
   G G G G G G G G G G G

Cluster 10

Sequences (4)

1. Sequence 5
   A A A A A A A A A A A A A A A A A A A
   G                             G G G G

2. Sequence 17
   A A A A A A A A A A A A A A A A A A A
   G G G G G

3. Sequence 28
   A A A A A A A A A A A A A A A A A A A
   G G G G G J J J J J J J J J J J J J J

4. Sequence 41
   A A A A A A A A A A A A A A A A A A A
   G J J J J J J J J J J J J J J J J J J
```

It can be seen that sequence 38 (cluster 8) has more sociological similarity to sequence 28 from cluster 10 than with the sequences in its current cluster. Additionally, this is evident for sequence 70 (cluster 8) and sequence 17 (cluster 10). The wrong allocation of sequences to some clusters suggests that the MCSA algorithms is less capable of yielding similarity-values that are based on common sociological considerations. It is not immediately clear what causes this behaviour, but the MCSA algorithm has yielded unexpected results throughout this paper, and it is evident that the MCSA algorithm requires some work to find and fix the causes of this sociological infidelities.

All in all, the OM, Dynamic Hamming and Mulder algorithm seem to be at a generally acceptable level with regard to generating sociologically expected results. OM could improve in some cases, but is especially the MCSA algorithm that produces unexpected results.

Third indicator : density of clusters

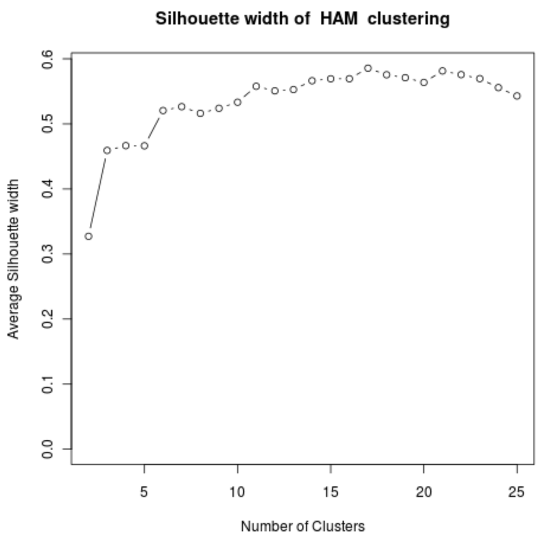
This section builds on top of the previous section. Where the previous section has shown the comparative level of sociologically expected results, this section shows how confident the different algorithms are when yielding these results. A more confident algorithm that still produces the same sociologically expected results as other algorithms can be said to have stronger 'feeling' for grouping sociologically similar sequences together. On the other hand, when the algorithm is not confident about its choice, created clusters become more based on luck than on purposeful determination.

The way to measure the confidence of the algorithms is to analyse the 'compactness' of the different clusters. If one cluster is very compact, i.e. when the sequences in the cluster have very high similarity-values to each other but not to sequences outside the cluster, the algorithm was confident that exactly those sequences had to be grouped with one another and not with different sequences.

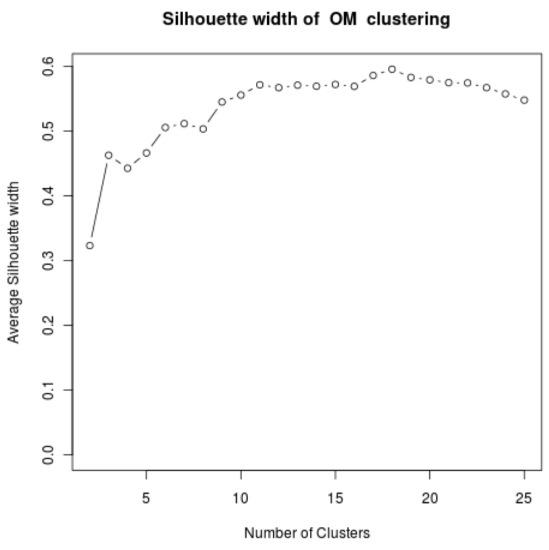
Note that confidence is only good when the subject of that confidence turns out to be good as well. In other words, higher confidence for algorithms that produce sociologically valid results is desirable, but high confidence for algorithms that produce sociologically invalid results is not desirable. In terms of the discussed algorithms, this would imply that for the OM, Hamming and Mulder algorithms, it is hoped to find a high confidence. However, for the MCSA algorithms, it is hoped to find a lower confidence, since a low confidence would indicate that the MCSA does not have strong determinants for its invalid outcomes, and consequently it is more open for improvement than when its current assumptions would be in the way.

This paper uses the 'silhouette-index' (Rousseeuw, 1987) to measure the compactness of clusters. The silhouette-index compares the similarity of sequences *within* a cluster with the distance to the nearest *other* cluster (Brock et al., 2007). Both a high within-cluster similarity and a low between-cluster similarity yield a higher silhouette-index. Similarity, a low within-cluster similarity and a high between-cluster similarity yield a lower silhouette-index. The indexes for the different methods are as follows.

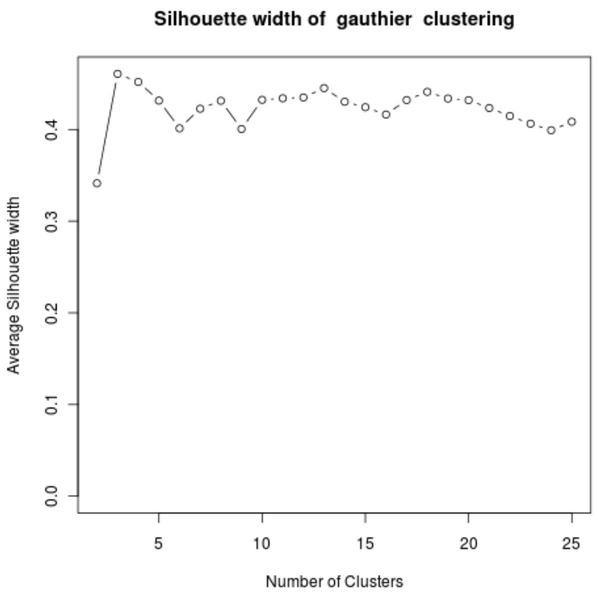
Dynamic Hamming



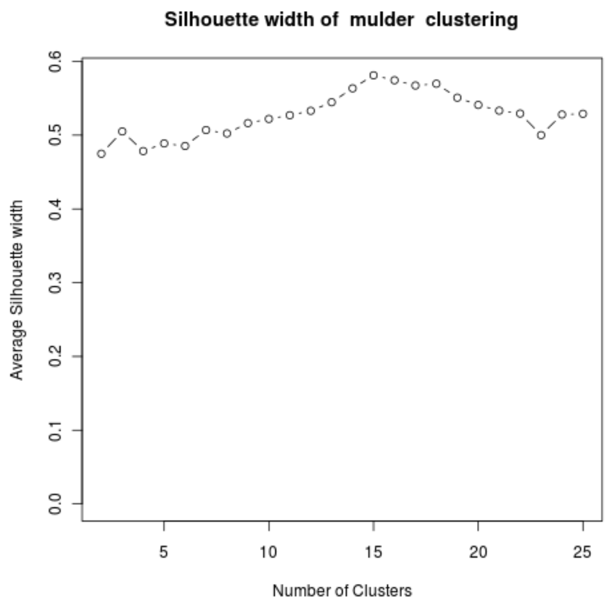
OM



MCSA



Mulder



Of all the algorithms, it is easily seen that the MCSA algorithm has the lowest confidence concerning its grouping. Its silhouette-index revolves around 0,4 but never really gets higher. On the other hand, the Hamming, OM and Mulder algorithm have an overall higher silhouette-index that additionally increases when the number of clusters goes up, until an optimal point between 15 and 20 clusters. The Mulder-algorithm seems to perform best with the highest average index, specifically coming from high scores when using small amounts of clusters.

This seems to correlate with earlier findings. The algorithms with highest sociological validity seem to have highest confidence, while the MCSA algorithm with the lower sociological validity also has the lowest confidence about its results.

Considering all indicators

Different indicators have shown the extent to which the different algorithms yield sociologically meaningful results and how confident the algorithms are while doing so. Taking into account all three indicators, it is seen that of all algorithms the Mulder algorithm seems to perform relatively best for this data-set. The Hamming Distance does also confidently yield sociologically valid clusters, but it could, just like OM and MCSA, not tease out the changes between 'single' and 'in a relationship' in a representative sequence, and it was already helped with a data-set containing sequences of equal length. The OM algorithm did also produce sociologically valid results, but its placement of some sequences in certain groups was unexpected. The MCSA method finally is sociologically performing the worst. Clusters seem to have no sociological foundation behind them, and the confidence over its groups is more than 25% lower than the confidence of the other algorithms.

Discussion

Considering the above results, did this paper answer its initial question of how the Mulder algorithm is more useful than other algorithms? One might be inclined to say yes, but there's some issues still unaddressed. First, the algorithm is not conclusively benchmarked on speed in comparison with the other algorithms. This was inherently impossible due to the differences in coding language between the current Mulder software and the software of the other algorithms. Consequently, any benchmark would not only test the isolated algorithm, but also the speed of the different languages, compilers, etc. Thus, the current implementation hinders a conclusive objective comparison. Additionally, it also hinders general adoption since the algorithm is not available within any major package, but is rather a relatively slow separate piece of software. Another issue is that of alignment. Currently, the algorithm is only concerned with relative distances between elements (i.e. A is 5 before B) but generally not with absolute time-positions, unless it is for the time-dependent substitution matrices. It needs more investigation to find out how absolute time-positions are (potentially) useful, and how the Mulder algorithm can be made aware of them. Another consequence of the emphasis on relative distances between elements is that the Mulder algorithm can currently not distinguish between the beginning and the end of a sequence. This is especially relevant when one wants to deal somehow with terminated or unfinished sequences, i.e. sequences that might grow (or might have grown) longer than their current state. It is evident that one needs to know the extent to which one sequence is the start of another sequence in order to do any projection about potential or future matching. A last issue is that the Mulder algorithm has no formal notion of insertion or deletion, so that the support for context-dependent matching can only be partial. This might not have been a problem for the cases in this paper, but there might be situations where the context of an element plays an important role for the value of characteristics that the element is present in. The Mulder algorithm will currently fail to take that into account.

Conclusion and recommendations

Still, despite those edge-cases and lack of comparative benchmarking on speed, the Mulder algorithm has shown to be a very useful algorithm that can yield sociologically valid similarity values for pairs of sequences in many situations. Obviously, examples as used in this paper can not prove anything and are never complete, but they clearly indicate that the Mulder algorithm can be a promising new addition to the methods that sociologists can use for their analyses.

Not only does the Mulder algorithm yield sociologically expected results in basic situations (chapter 6), it also outperforms existing algorithms with regard to the valuation of order and the support for time-dependent substitution matrices. Moreover, it does so while supporting both single-channel and multi-channel data, as well as sequences of unequal length. Finally, tests on existing data-sets show that the Mulder algorithm produces sociologically expected results for real-life data, and that the algorithm is confident in doing so.

Two other positive points of the Mulder algorithm that have only been touched briefly in this paper are also worth mentioning.

First, similarity-values always range between 0 and 1. This is of great help to easily interpret the amount of similarity, as it gives the researcher a frame of reference in terms of a lower and upper limit while also relating to the common notion of percentages.

Second, the Mulder algorithm supports importancy values. This enables researchers to define how certain (types of) characteristics are more important than others. Since such functionality has not been proposed in literature, this opens up many new possibilities for doing research, and the coming years have to show whether the field of social sciences finds this useful or not.

Regarding the current problems with regard to speed, these can be solved by rewriting the algorithm in a compiled language (contrary to the current interpreted language) and implementing other optimisations. If the algorithm is written inside an existing statistical package such as R, this would additionally open up ample possibilities for other researchers to try out the method and improve upon it.

It is the hope of the author that constructive scientific discussion leads to many useful applications and improvements for the Mulder method. Specifically, the method needs more testing with various types of data in order to determine its usefulness in the most diverse cases. This also includes situations where context-dependent matching is important or where the problem of unfinished sequences comes up. Indeed, it is only through repeated and extensive testing that there continue to be possibilities for amelioration of functionality and applicability, making for a method that, even though it will never be perfect, helps researchers in the best and most useful way.

References

Aassve et al. 2007. Strings of Adulthood: Analyzing work-family trajectories using sequence analysis. *European Journal of Population.*

Abbott A. and S. DeViney . 1992. The Welfare State as Transnational Event: Evidence from Sequences of Policy Adoption . *Social Science History .* Vol. 16, No. 2 (Summer, 1992), pp. 245-274.

Abbott, A. and J. Forrest. 1986. Optimal Matching Methods for Historical Sequences. *Journal of Interdisciplinary History*, Vol. 16, No. 3, pp. 471-494.

Abbott, A. and A. Hrycak. 1990. Measuring resemblance in sequence data: An optimal matching analysis of musicians' careers. *American journal of sociology.* Volume 96(1), pp. 144.

Abbott, A. and A. Tsay. 2000. Sequence Analysis and Optimal Matching Methods in Sociology: Review and Prospect . *Sociological Methods Research.* Volume 29(1), pp. 3-33.

Abbott, A. 2000. Reply to Levine and Wu. *Sociological Methods & Research.* Volume 29(1), pp. 65-76.

Abbott, A. 1995. A Comment on 'Measuring the Agreement between Sequences.' *Sociological Methods & Research* . Volume 24(2), pp. 232-43.

Aisenbrey, S. and A. E. Fasang. 2010. New Life for Old Ideas: The "Second Wave" of Sequence Analysis Bringing the "Course" Back Into the Life Course. *Sociological Methods & Research.* Volume 38(3), pp. 420-462.

Blair-Loy, M. 1999. Career Patterns of Executive Women in Finance: An Optimal Matching Analysis. *American Journal of Sociology.* Volume 104, pp. 46-97.

Brzinsky-Fay, C. and U. Kohler . 2010. New Developments in Sequence Analysis. *Sociological Methods & Research.* Volume 38, pp. 359-364.

Brzinsky-Fay, C. 2007. "Lost in Transition? Labour Market Entry Sequences of School Leavers in Europe." *European Sociological Review.* Volume 23(4), pp. 409–422.

Dijkstra, W. and T. Taris. 1995. "Measuring the Agreement between Sequences." *Sociological Methods & Research.* Volume 24(2), pp. 214-231.

Elzinga, C. H. 2003. Sequence Similarity: A Non-aligning Technique. *Sociological Methods & Research.* Volume 32(1), pp. 3-29.

Elzinga, C. H. 2005. Combinatorial Representation of Token Sequences. *Journal of Classification .* Volume 22, pp. 87–118.

Elzinga, C. H. 2007. Sequence Analysis: Metric Representations of Categorical Time Series. Manuscript, Department of Social Science Research Methods, Vrije Universiteit, Amsterdam.

Elzinga, C. H. and A. C. Liefbroer (2007). "De-standardization of Family-Life Trajectories of Young Adults: A Cross-National Comparison Using Sequence Analysis." *European Journal of Population* 23, pp. 225-50.

Gabadinho, A., G. Ritschard, N. S. Müller and M. Studer (2011). Analyzing and Visualizing State Sequences in R with TraMineR. Journal of Statistical Software 40(4), 1-37.

Gabadinho, A., G. Ritschard, M. Studer and N. S. Müller (2009). Mining Sequence Data in R with the TraMineR package: A user's guide. Department of Econometrics and Laboratory of Demography, University of Geneva

Garnsey, E.,, E. Stam, P. Heffernan and O. Hugo . 2003. "New Firm Growth: Exploring Processes and Paths " Erim report series research in management

Gauthier, J.-A., E.D Widmer, P. Bucher and C. Notredame. 2009. How much does it cost? Optimization of costs in sequence analysis of social science data. *Sociological Methods and Research.* Volume 38 (1), pp. 197-231.

Gauthier, J.-A., Widmer, E. D., Bucher, P. and Notredame, C. 2010. Multichannel sequence analysis applied to social science data. *Sociological Methodology*. Volume 40, pp. 1–38.

Halpin, B. 2010. Optimal Matching Analysis and Life-Course Data: The Importance of Duration . *Sociological Methods & Research.* Volume 38 (3), pp. 365-388.

Han, S.-K., and P. Moen. 1999. Clocking Out: Temporal Patterning of Retirement. *American Journal of Sociology.* Volume 105 (1), pp. 191–236.

Hollister, M. 2009. Is Optimal Matching Suboptimal? *Sociological Methods & Research .* Volume 38 (2), pp. 235–264 .

Lesnard, L. 2010. Setting Cost in Optimal Matching to Uncover Contemporaneous Socio-Temporal Patterns . *Sociological Methods & Research.* Volume 38 (3), pp. 389-419.

Levine, J. H. 2000. But What Have You Done for Us Lately? Commentary on Abbott and Tsay. *Sociological Methods & Research.* Volume 29 (1), pp. 34-40.

Martin, P., I. Schoon, and A. Ross. (2008). "Beyond Transitions: Applying Optimal Matching Analysis to Life Course Research." *International Journal of Social Research Methodology* Volume 11, pp. 179-99.

Massoni, S., M. Olteneau and P. Rousset. 2009. Career-Path Analysis Using Optimal Matching and Self-Organizing Maps. *WSOM* 2009, LNCS 5629, pp. 154–162.

Needleman, S. B. and Wunsch, C. D. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology.* Volume 48 (3), pp. 443–453.

Pollock, G. 2007. Holistic Trajectories: A Study of Combined Employment, Housing and Family Careers by Using Multiple-Sequence Analysis. *Journal of the Royal Statistical Society: Series A (Statistics in Society).* Volume 170 (1), pp. 167-83.

Rousseeuw, P. J., 1987. Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Computational and Applied Mathematics* .Volume 20, pp. 53–65.

Sankoff, D. 1972. Matching sequences under deletion/insertion constraint. *Proc Natl Acad Sci USA.* Volume 69, pp. 4–6.

Scherer, S. 2001. Early Career Patterns: A Comparison of Great Britain and West Germany. *European Sociological Review* . Volume 17, pp. 119-44.

Ven, van de, A. H. and M.S. Poole. 1990. Methods for Studying Innovation Development in the Minnesota Innovation Research Program. *Organization Science,* Volume 1, No. 3, Special Issue: Longitudinal Field Research Methods for Studying Processes of Organizational Change (1990), pp. 313-335.

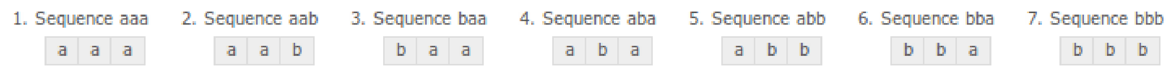
Wiggins, R. D., C. Erzberger, M. Hyde, P. Higgs, and D. Blane. 2007. Optimal Matching Analysis Using Ideal Types to Describe the Lifecourse: An Illustration of How Histories of Work, Partnerships and Housing Relate to Quality of Life in Early Old Age. *International Journal of Social Research Methodology*. Volume 10, pp. 259-78.

Wu, L. L. 2000. Some Comments on "Sequence Analysis and Optimal Matching Methods in Sociology: Review and Prospect" *Sociological methods and research.* Volume 29(1), pp. 41-64.

Appendix 1 : Full results for validation-chapter

Full validation results for the "All best matches" implementation of Mulder

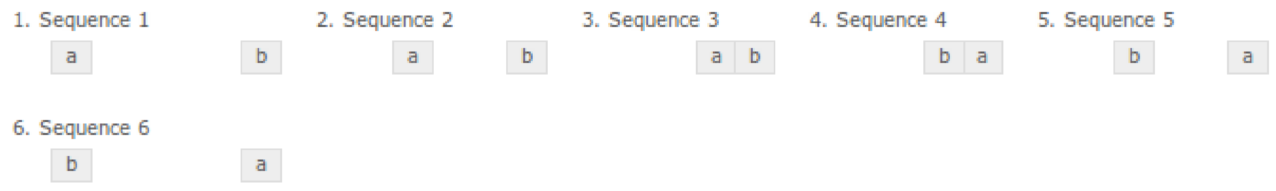
Sequences (7)



Similarities

Sequence aaa	Sequence aab	Sequence baa	Sequence aba	Sequence abb	Sequence bba	Sequence bbb
aaa: 1	aaa: 0.74	aaa: 0.74	aaa: 0.73	aaa: 0.54	aaa: 0.54	aaa: 0
aab: 0.74	aab: 1	aab: 0.73	aab: 0.91	aab: 0.98	aab: 0.71	aab: 0.54
baa: 0.74	baa: 0.73	baa: 1	baa: 0.91	baa: 0.71	baa: 0.98	baa: 0.54
aba: 0.73	aba: 0.91	aba: 0.91	aba: 1	aba: 0.9	aba: 0.9	aba: 0.54
abb: 0.54	abb: 0.98	abb: 0.71	abb: 0.9	abb: 1	abb: 0.73	abb: 0.74
bba: 0.54	bba: 0.71	bba: 0.98	bba: 0.9	bba: 0.73	bba: 1	bba: 0.74
bbb: 0	bbb: 0.54	bbb: 0.54	bbb: 0.54	bbb: 0.74	bbb: 0.74	bbb: 1

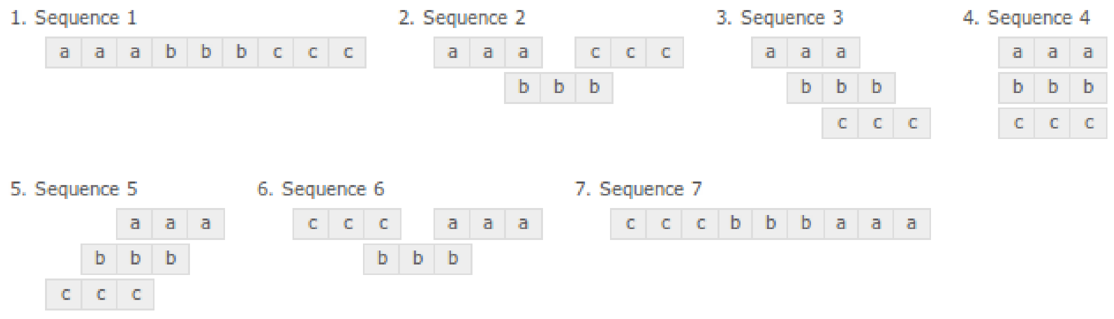
Sequences (6)



Similarities

Sequence 1	Sequence 2	Sequence 3	Sequence 4	Sequence 5	Sequence 6
1: 1	1: 0.97	1: 0.93	1: 0.73	1: 0.7	1: 0.67
2: 0.97	2: 1	2: 0.96	2: 0.75	2: 0.71	2: 0.7
3: 0.93	3: 0.96	3: 1	3: 0.78	3: 0.75	3: 0.73
4: 0.73	4: 0.75	4: 0.78	4: 1	4: 0.96	4: 0.93
5: 0.7	5: 0.71	5: 0.75	5: 0.96	5: 1	5: 0.97
6: 0.67	6: 0.7	6: 0.73	6: 0.93	6: 0.97	6: 1

Sequences (7)



Similarities

Sequence 1	Sequence 2	Sequence 3	Sequence 4	Sequence 5	Sequence 6	Sequence 7
1: 1	1: 1	1: 0.97	1: 0.91	1: 0.83	1: 0.73	1: 0.59
2: 1	2: 1	2: 0.99	2: 0.96	2: 0.91	2: 0.82	2: 0.73
3: 0.97	3: 0.99	3: 1	3: 0.99	3: 0.97	3: 0.91	3: 0.83
4: 0.91	4: 0.96	4: 0.99	4: 1	4: 0.99	4: 0.96	4: 0.91
5: 0.83	5: 0.91	5: 0.97	5: 0.99	5: 1	5: 0.99	5: 0.97
6: 0.73	6: 0.82	6: 0.91	6: 0.96	6: 0.99	6: 1	6: 1
7: 0.59	7: 0.73	7: 0.83	7: 0.91	7: 0.97	7: 1	7: 1

Sequences (5)

1. Sequence 1
```
a a a a a
```
2. Sequence 2
```
a a a a a a
```
3. Sequence 3
```
a a a a a b
```
4. Sequence 4
```
a a a a a a a a a a
```
5. Sequence 5
```
a a a a a b b b b b
```

Similarities

Sequence 1	Sequence 2	Sequence 3	Sequence 4	Sequence 5
1: 1	1: 1	1: 0.83	1: 0.99	1: 0.43
2: 1	2: 1	2: 0.86	2: 0.99	2: 0.47
3: 0.83	3: 0.86	3: 1	3: 0.91	3: 0.99
4: 0.99	4: 0.99	4: 0.91	4: 1	4: 0.63
5: 0.43	5: 0.47	5: 0.99	5: 0.63	5: 1

Sequences (3)

1. Sequence 1
```
b b b a b b a a a a
```
2. Sequence 2
```
b b b a a b b a a a a
```
3. Sequence 3
```
b b b a b b b a a a a
```

Similarities

Sequence 1	Sequence 2	Sequence 3
1: 1	1: 1	1: 1
2: 1	2: 1	2: 1
3: 1	3: 1	3: 1

Sequences (5)

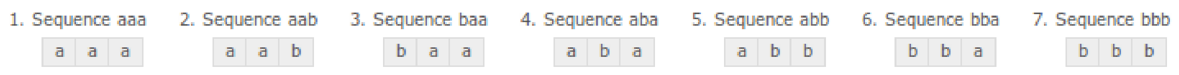
1. Sequence 1
```
a a a a a a a a a a a a a a       a a a a a
      b b     d d d d d d d
    c c c c c c c c c c c c c c c
```
2. Sequence 2
```
a a a a a a a a a a a a a a       a a a a a
    c c c c c c c c c c c c c c c
              d d d d d d d
```
3. Sequence 3
```
a a a a a a a a a a a a a a       a a a a a
              d d d d d d d
```
4. Sequence 4
```
a a a a a a a a a a a a a a a a a a a a a a
```
5. Sequence 5
```
a a a a a       a a a a a a a a a a a a a a
                d d d d d d d     b b
          c c c c c c c c c c c c c c c
```

Similarities

Sequence 1	Sequence 2	Sequence 3	Sequence 4	Sequence 5
1: 1	1: 0.95	1: 0.54	1: 0.36	1: 0.99
2: 0.95	2: 1	2: 0.58	2: 0.39	2: 0.95
3: 0.54	3: 0.58	3: 1	3: 0.73	3: 0.54
4: 0.36	4: 0.39	4: 0.73	4: 1	4: 0.36
5: 0.99	5: 0.95	5: 0.54	5: 0.36	5: 1

Full validation results for the "multi-match penalty" implementation

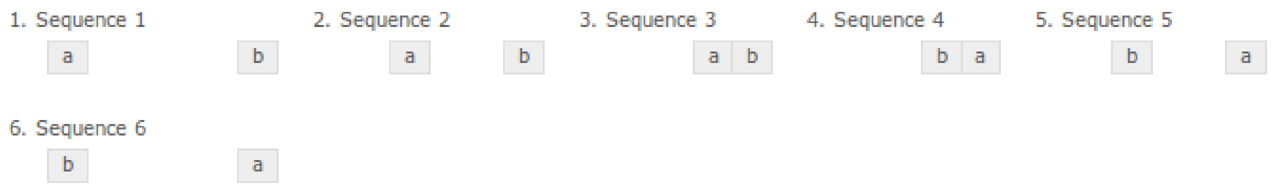
Sequences (7)



Similarities

Sequence aaa	Sequence aab	Sequence baa	Sequence aba	Sequence abb	Sequence bba	Sequence bbb
aaa: 1	aaa: 0.45	aaa: 0.45	aaa: 0.45	aaa: 0.27	aaa: 0.27	aaa: 0
aab: 0.45	aab: 1	aab: 0.69	aab: 0.73	aab: 0.66	aab: 0.34	aab: 0.27
baa: 0.45	baa: 0.69	baa: 1	baa: 0.83	baa: 0.34	baa: 0.66	baa: 0.27
aba: 0.45	aba: 0.73	aba: 0.83	aba: 1	aba: 0.41	aba: 0.41	aba: 0.27
abb: 0.27	abb: 0.66	abb: 0.34	abb: 0.41	abb: 1	abb: 0.71	abb: 0.41
bba: 0.27	bba: 0.34	bba: 0.66	bba: 0.41	bba: 0.71	bba: 1	bba: 0.45
bbb: 0	bbb: 0.27	bbb: 0.27	bbb: 0.27	bbb: 0.41	bbb: 0.45	bbb: 1

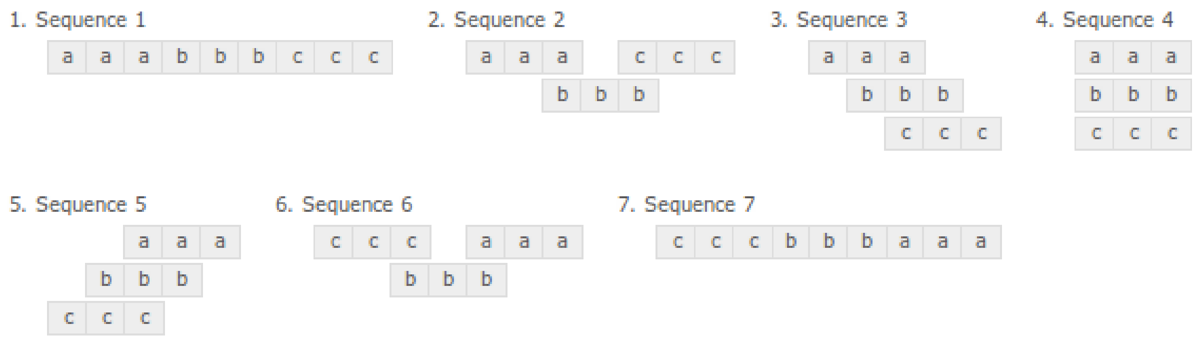
Sequences (6)



Similarities

Sequence 1	Sequence 2	Sequence 3	Sequence 4	Sequence 5	Sequence 6
1: 1	1: 0.97	1: 0.93	1: 0.73	1: 0.7	1: 0.67
2: 0.97	2: 1	2: 0.96	2: 0.75	2: 0.71	2: 0.7
3: 0.93	3: 0.96	3: 1	3: 0.78	3: 0.75	3: 0.73
4: 0.73	4: 0.75	4: 0.78	4: 1	4: 0.96	4: 0.93
5: 0.7	5: 0.71	5: 0.75	5: 0.96	5: 1	5: 0.97
6: 0.67	6: 0.7	6: 0.73	6: 0.93	6: 0.97	6: 1

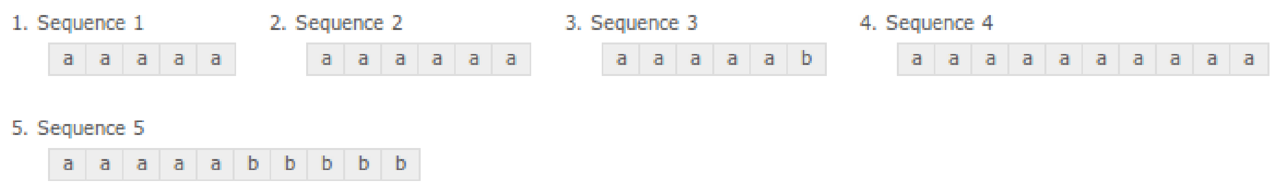
Sequences (7)



Similarities

Sequence 1	Sequence 2	Sequence 3	Sequence 4	Sequence 5	Sequence 6	Sequence 7
1: 1	1: 0.76	1: 0.71	1: 0.63	1: 0.59	1: 0.54	1: 0.49
2: 0.76	2: 1	2: 0.74	2: 0.69	2: 0.63	2: 0.58	2: 0.55
3: 0.71	3: 0.74	3: 1	3: 0.75	3: 0.73	3: 0.64	3: 0.6
4: 0.63	4: 0.69	4: 0.75	4: 1	4: 0.75	4: 0.7	4: 0.64
5: 0.59	5: 0.63	5: 0.73	5: 0.75	5: 1	5: 0.74	5: 0.72
6: 0.54	6: 0.58	6: 0.64	6: 0.7	6: 0.74	6: 1	6: 0.76
7: 0.49	7: 0.55	7: 0.6	7: 0.64	7: 0.72	7: 0.76	7: 1

Sequences (5)

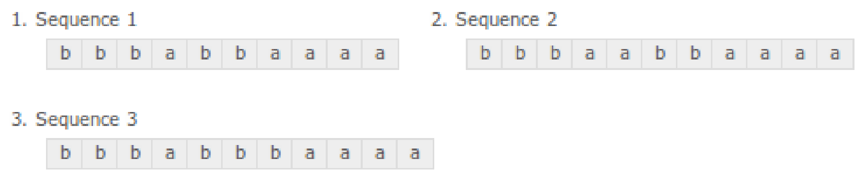


1. Sequence 1
```
a a a a a
```
2. Sequence 2
```
a a a a a a
```
3. Sequence 3
```
a a a a a b
```
4. Sequence 4
```
a a a a a a a a a a
```
5. Sequence 5
```
a a a a a b b b b b
```

Similarities

Sequence 1	Sequence 2	Sequence 3	Sequence 4	Sequence 5
1: 1	1: 0.78	1: 0.69	1: 0.49	1: 0.21
2: 0.78	2: 1	2: 0.67	2: 0.54	2: 0.18
3: 0.69	3: 0.67	3: 1	3: 0.45	3: 0.7
4: 0.49	4: 0.54	4: 0.45	4: 1	4: 0.31
5: 0.21	5: 0.18	5: 0.7	5: 0.31	5: 1

Sequences (3)

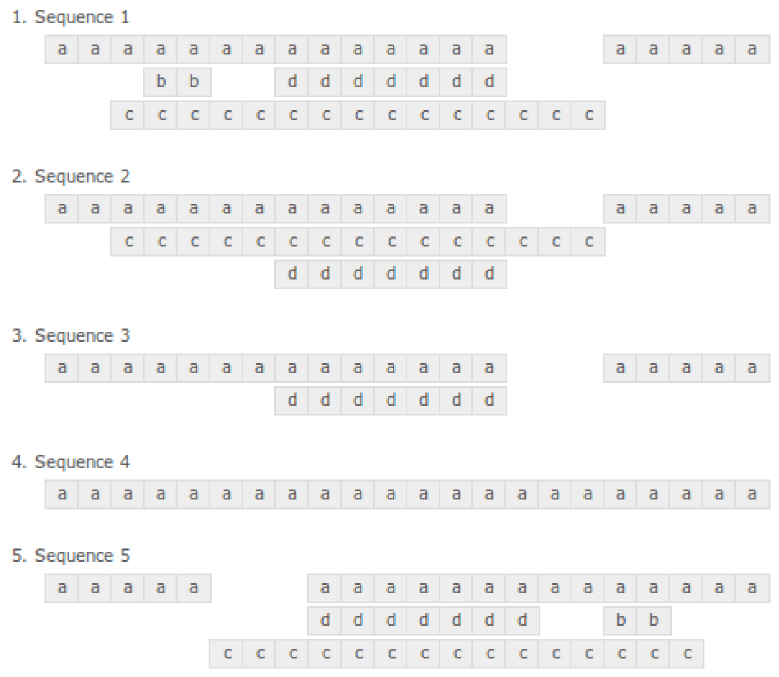


1. Sequence 1
```
b b b a b b a a a a
```
2. Sequence 2
```
b b b a a b b a a a a
```
3. Sequence 3
```
b b b a b b b a a a a
```

Similarities

Sequence 1	Sequence 2	Sequence 3
1: 1	1: 0.8	1: 0.85
2: 0.8	2: 1	2: 0.83
3: 0.85	3: 0.83	3: 1

Sequences (5)



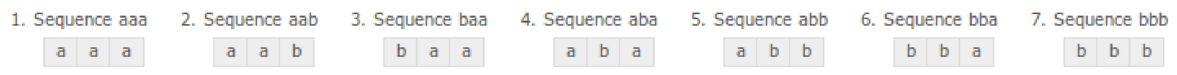
1. Sequence 1
```
a a a a a a a a a a a a a a       a a a a a
      b b       d d d d d d d
    c c c c c c c c c c c c c c c
```
2. Sequence 2
```
a a a a a a a a a a a a a a       a a a a a
    c c c c c c c c c c c c c c c
              d d d d d d d
```
3. Sequence 3
```
a a a a a a a a a a a a a a       a a a a a
              d d d d d d d
```
4. Sequence 4
```
a a a a a a a a a a a a a a a a a a a a a a
```
5. Sequence 5
```
a a a a a       a a a a a a a a a a a a a a
                d d d d d d d     b b
          c c c c c c c c c c c c c c c
```

Similarities

Sequence 1	Sequence 2	Sequence 3	Sequence 4	Sequence 5
1: 1	1: 0.9	1: 0.27	1: 0.14	1: 0.89
2: 0.9	2: 1	2: 0.29	2: 0.15	2: 0.83
3: 0.27	3: 0.29	3: 1	3: 0.45	3: 0.26
4: 0.14	4: 0.15	4: 0.45	4: 1	4: 0.14
5: 0.89	5: 0.83	5: 0.26	5: 0.14	5: 1

Full validation results for the "Homogeneity" implementation of Mulder

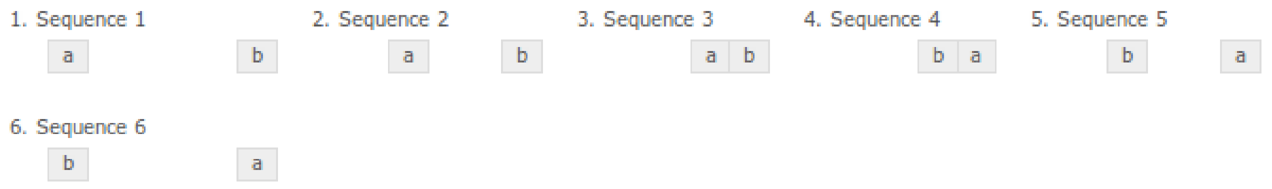
Sequences (7)



Similarities

Sequence aaa	Sequence aab	Sequence baa	Sequence aba	Sequence abb	Sequence bba	Sequence bbb
aaa: 1	aaa: 0.43	aaa: 0.43	aaa: 0.43	aaa: 0.13	aaa: 0.13	aaa: 0
aab: 0.43	aab: 1	aab: 0.72	aab: 0.79	aab: 0.66	aab: 0.43	aab: 0.13
baa: 0.43	baa: 0.72	baa: 1	baa: 0.79	baa: 0.43	baa: 0.66	baa: 0.13
aba: 0.43	aba: 0.79	aba: 0.79	aba: 1	aba: 0.47	aba: 0.47	aba: 0.13
abb: 0.13	abb: 0.66	abb: 0.43	abb: 0.47	abb: 1	abb: 0.72	abb: 0.43
bba: 0.13	bba: 0.43	bba: 0.66	bba: 0.47	bba: 0.72	bba: 1	bba: 0.43
bbb: 0	bbb: 0.13	bbb: 0.13	bbb: 0.13	bbb: 0.43	bbb: 0.43	bbb: 1

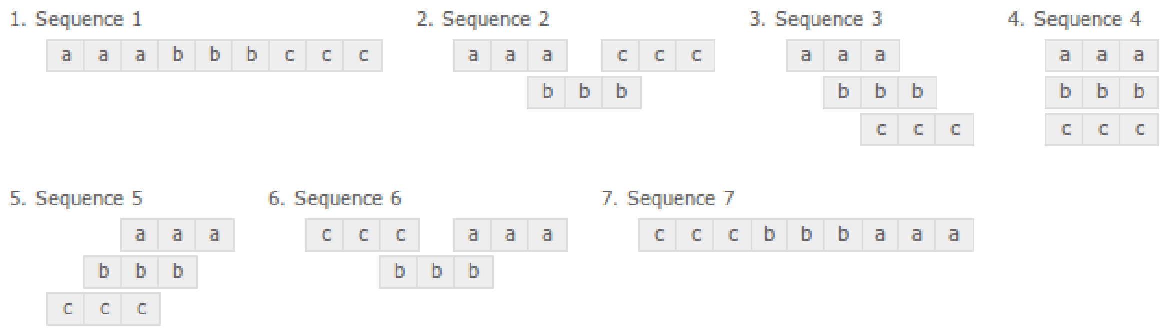
Sequences (6)



Similarities

Sequence 1	Sequence 2	Sequence 3	Sequence 4	Sequence 5	Sequence 6
1: 1	1: 0.98	1: 0.97	1: 0.87	1: 0.85	1: 0.56
2: 0.98	2: 1	2: 0.98	2: 0.88	2: 0.85	2: 0.85
3: 0.97	3: 0.98	3: 1	3: 0.89	3: 0.88	3: 0.87
4: 0.87	4: 0.88	4: 0.89	4: 1	4: 0.98	4: 0.97
5: 0.85	5: 0.85	5: 0.88	5: 0.98	5: 1	5: 0.98
6: 0.56	6: 0.85	6: 0.87	6: 0.97	6: 0.98	6: 1

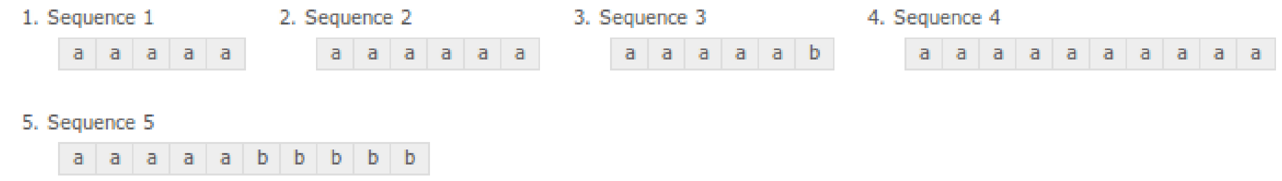
Sequences (7)



Similarities

Sequence 1	Sequence 2	Sequence 3	Sequence 4	Sequence 5	Sequence 6	Sequence 7
1: 1	1: 0.74	1: 0.56	1: 0.44	1: 0.41	1: 0.39	1: 0.37
2: 0.74	2: 1	2: 0.74	2: 0.56	2: 0.49	2: 0.45	2: 0.39
3: 0.56	3: 0.74	3: 1	3: 0.75	3: 0.63	3: 0.49	3: 0.41
4: 0.44	4: 0.56	4: 0.75	4: 1	4: 0.75	4: 0.56	4: 0.44
5: 0.41	5: 0.49	5: 0.63	5: 0.75	5: 1	5: 0.74	5: 0.56
6: 0.39	6: 0.45	6: 0.49	6: 0.56	6: 0.74	6: 1	6: 0.74
7: 0.37	7: 0.39	7: 0.41	7: 0.44	7: 0.56	7: 0.74	7: 1

Sequences (5)

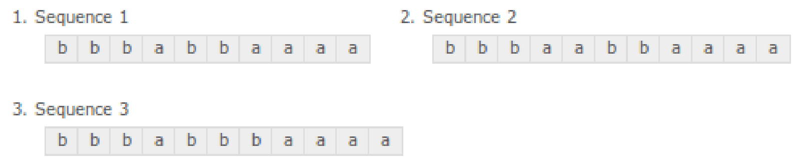


1. Sequence 1
```
a a a a a
```
2. Sequence 2
```
a a a a a a
```
3. Sequence 3
```
a a a a a b
```
4. Sequence 4
```
a a a a a a a a a a
```
5. Sequence 5
```
a a a a a b b b b b
```

Similarities

Sequence 1	Sequence 2	Sequence 3	Sequence 4	Sequence 5
1: 1	1: 0.83	1: 0.76	1: 0.43	1: 0.31
2: 0.83	2: 1	2: 0.66	2: 0.55	2: 0.29
3: 0.76	3: 0.66	3: 1	3: 0.38	3: 0.55
4: 0.43	4: 0.55	4: 0.38	4: 1	4: 0.22
5: 0.31	5: 0.29	5: 0.55	5: 0.22	5: 1

Sequences (3)

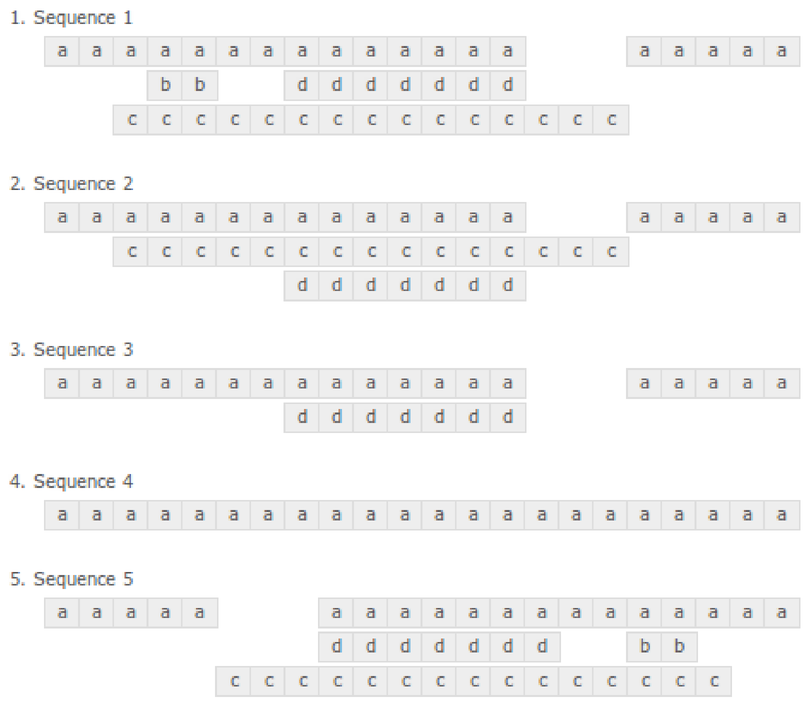


1. Sequence 1
```
b b b a b b a a a a
```
2. Sequence 2
```
b b b a a b b a a a a
```
3. Sequence 3
```
b b b a b b b a a a a
```

Similarities

Sequence 1	Sequence 2	Sequence 3
1: 1	1: 0.84	1: 0.89
2: 0.84	2: 1	2: 0.88
3: 0.89	3: 0.88	3: 1

Sequences (5)



1. Sequence 1
```
a a a a a a a a a a a a a a       a a a a a
      b b
              d d d d d d d
    c c c c c c c c c c c c c c c
```
2. Sequence 2
```
a a a a a a a a a a a a a a       a a a a a
    c c c c c c c c c c c c c c c
              d d d d d d d
```
3. Sequence 3
```
a a a a a a a a a a a a a a       a a a a a
              d d d d d d d
```
4. Sequence 4
```
a a a a a a a a a a a a a a a a a a a a a a
```
5. Sequence 5
```
a a a a a       a a a a a a a a a a a a a a
                d d d d d d d     b b
          c c c c c c c c c c c c c c c
```

Similarities

Sequence 1	Sequence 2	Sequence 3	Sequence 4	Sequence 5
1: 1	1: 0.93	1: 0.41	1: 0.21	1: 0.88
2: 0.93	2: 1	2: 0.45	2: 0.23	2: 0.87
3: 0.41	3: 0.45	3: 1	3: 0.54	3: 0.38
4: 0.21	4: 0.23	4: 0.54	4: 1	4: 0.21
5: 0.88	5: 0.87	5: 0.38	5: 0.21	5: 1

Appendix 2 : Distance matrices for the multi-channel comparison

Mulder

	1	2	3	4	5	6	7	8
1	0	0.50	0.60	0.65	0.64	0.70	0.76	0.91
2	0.50	0	0.64	0.61	0.56	0.74	0.97	0.85
3	0.60	0.64	0	0.65	0.76	0.66	0.89	0.89
4	0.65	0.61	0.65	0	0.26	0.43	0.89	0.81
5	0.64	0.56	0.76	0.26	0	0.30	0.91	0.91
6	0.70	0.74	0.66	0.43	0.30	0	0.84	0.98
7	0.76	0.97	0.89	0.89	0..91	0.84	0	1
8	0.91	0.85	0.89	0.81	0.91	0.98	1	0

OM

	1	2	3	4	5	6	7	8
1	0	3.75	3	2.25	2.25	3	1.5	2.75
2	3.75	0	2	3.5	4.25	4.25	3.75	4.25
3	3	2	0	3	3.75	2.75	1.75	2.25
4	2.25	3.5	3	0	1.25	2.5	3	3
5	2.25	4.25	3.75	1.25	0	1.75	2.75	2.75
6	3	4.25	2.75	2.5	1.75	0	2	1.75
7	1.5	3.75	1.75	3	2.75	2	0	1.25
8	2.75	4.25	2.25	3	2.75	1.75	1.25	0

Dynamic Hamming

	1	2	3	4	5	6	7	8
1	0	3.75	3.25	2.25	2.5	3	1.5	2.75
2	3.75	0	2	3.5	4.25	4.25	3.75	4.25
3	3.25	2	0	3	3.75	3	1.75	2.25
4	2.25	3.5	3	0	1.25	2.5	3	3
5	2.5	4.25	3.75	1.25	0	1.75	2.75	2.75
6	3	4.25	3	2.5	1.75	0	2	1.75
7	1.5	3.75	1.75	3	2.75	2	0	1.25
8	2.75	4.25	2.25	3	2.75	1.75	1.25	0

MCSA

	1	2	3	4	5	6	7	8
1	0	6700	3400	3400	4500	1700	6700	6700
2	6700	0	3400	3400	2300	5000	10000	3400
3	3400	3400	0	0	0	1700	6700	6700
4	3400	3400	0	0	1200	1700	6700	6700
5	4500	2300	0	1200	0	1700	6700	6700
6	1700	5000	1700	1700	1700	0	6700	6700
7	6700	10000	6700	6700	6700		0	10000
8	6700	3400	6700	6700	6700	6700	10000	0