

Similarity Measures and Clustering of String Patterns

Ana Fred
Telecommunications Institute
Instituto Superior Técnico, Technical University of Lisbon,
E-mail: `afred@lx.it.pt`

Contents

1	Introduction	2
2	Basic Definitions	4
2.1	Similarity, Dissimilarity and Proximity	4
2.2	Metrics and Dissimilarity Properties	5
3	String Patterns and Proximity Measures	6
3.1	Measuring Similarity between Sequences	6
3.2	String Alignment and Structural Similarity: Examples	8
3.2.1	Names Data Set	8
3.2.2	Fractal Images	9
3.2.3	Recursive Structure Line Patterns	11
3.2.4	Email Messages	11
3.2.5	Contour Images	12
3.3	Dissimilarity based on String Editing Operations	13
3.4	Dissimilarity based on Error-Correcting Parsing	15
3.5	Grammar Complexity-based Dissimilarity	17
3.6	Minimum Code Length-based Similarity	19
4	Clustering of String Patterns	21
4.1	Clustering Algorithms Tested	21
4.1.1	Hierarchical Agglomerative Clustering	21
4.1.2	Sentence-to-Sentence Clustering	23
4.2	Experimental Results	24

4.2.1	Names Data Set	24
4.2.2	Fractal Images	25
4.2.3	Recursive Structure Line Patterns	26
4.2.4	Email Messages	27
4.2.5	Contour Images	27
4.3	Conclusions	29
4.4	Current Trends in Clustering	30

References

Abstract

Clustering is a powerful tool in revealing the intrinsic organization of data. A clustering of structural patterns consists of an unsupervised association of data based on the similarity of their structures and primitives. This chapter addresses the problem of structural clustering, and presents an overview of similarity measures used in this context. The distinction between string matching and structural resemblance is stressed. The hierarchical agglomerative clustering concept and a partitional approach are explored in a comparative study of several dissimilarity measures: minimum code length based measures; dissimilarity based on the concept of reduction in grammatical complexity; and error-correcting parsing.

1 Introduction

Consider a data set D consisting of N sequences of symbols (or strings), $D = \{s_1, \dots, s_N\}$, defined over the alphabet Σ , where $s_i = (x_1^i x_2^i \dots x_{L_i}^i)$ is a sequence of length L_i . The problem of sequence clustering is defined as follows: given the sample patterns in D , discover from the data a “natural” grouping of the sequences into K clusters, K being in general unknown.

The problem of clustering in multivariate feature spaces has been extensively studied [32, 33, 11, 66]. Many clustering algorithms have been proposed, each with its own approach for handling cluster validity [1, 27, 53, 21], number of clusters [40, 65], and structure imposed in the data [63, 44]. Directly using dissimilarity values or exploring point densities for the patterns, either emphasizing compactness or connectedness in feature space, leads to two main strategies for clustering: hierarchical methods and partitional methods. Partitional structure organizes patterns into a small number of clusters; a data partition is obtained as the result of an optimization process or by exploring local structure. Examples of techniques in this class

include mixture decomposition [47, 57, 14], non-parametric density estimation based methods [54], central clustering [2], square-error clustering [49], shape fitting approaches [63, 44], and geometrical approaches [26]. The K-means is a very popular algorithm in this category. Assuming *a priori* knowledge about the number of classes and based on the square-error criterion, it is a computationally efficient clustering technique that identifies hyper-spherical clusters [32, 11].

Hierarchical methods produce a nesting of data clusterings, that can be represented graphically as a dendrogram. Mostly inspired by graph theory [67], both agglomerative [32, 12, 21] and divisive approaches [5] have been attempted, the first starting with many clusters that are successively merged in accordance with inter cluster similarity, and the later working in the opposite direction. Variations of the algorithms can be obtained by the definition of a similarity measure between patterns and clusters [20]. The single link algorithm is one of the most popular methods in this class [32]. Data partitioning is usually obtained by setting a threshold on the dendrogram; cluster validity studies have also been proposed [10, 1] for the *a posteriori* analysis of structures, in order to evaluate the clustering results and define meaningful clusters.

When patterns are described in symbolic form as strings of characters, clustering algorithms found in the literature are typically extensions of conventional clustering methods [32, 23] by introducing dissimilarity measures between strings [20]. This is non trivial since the sequences can be of different lengths; also it is not always clear what is a meaningful similarity/dissimilarity measure for sequence comparison. Viewing similarity computation as a matching process [23, 3, 4, 51, 46, 8, 56], references [23, 43] present sentence-to-sentence clustering procedures based on the comparison of a candidate string with sentences in previously formed clusters (clustering based on a nearest-neighbor rule) or with cluster center strings (cluster center technique). String editing operations are there used in the transformation of strings to perform the matching. Following the string matching paradigm while modelling clusters' structure using grammars, error-correcting parsing and grammatical inference are combined in a clustering algorithm described in [23, 24]. Basically, it implements a nearest-neighbor rule, where sentences are compared, not directly with the patterns included in previously formed clusters, but with the best matching elements in the languages generated by the grammars inferred from the clusters' data. Also, using grammars for modelling clusters' structure, a distinct approach, based on the concept of minimum description, is proposed in [16]. Structural resemblance between patterns is assumed to reflect common rules of composi-

tion; a normalized reduction in grammar complexity obtained by associating patterns gives the measure of similarity underlying the hierarchical clustering algorithm proposed there. In [18] the search of common sub-patterns by means of Solomonoff's coding [62, 19] forms the basis of a clustering algorithm that defines similarity between patterns as a ratio of decrease in code length. Other model-based clustering techniques have been proposed in the context of continuous valued discrete time series, making use of Markov chain models [60] or hidden Markov models (HMM) [50, 50, 61].

In this chapter the problem of clustering of string patterns is addressed, the distinction between string matching and structural resemblance being stressed. The first part of this paper presents an overview and discussion of dissimilarity measures for string patterns. These are further analyzed and compared in the context of clustering, based on a set of test examples, in the second part of the chapter, followed by conclusions and current trends in clustering.

2 Basic Definitions

2.1 Similarity, Dissimilarity and Proximity

Let $s_1 = (x_1^1 x_2^1 \dots x_{L_1}^1)$, $s_2 = (x_1^2 x_2^2 \dots x_{L_2}^2)$ and $s_3 = (x_1^3 x_2^3 \dots x_{L_3}^3)$ be strings defined over the alphabet Σ , and let $|s_i| = L_i$ denote the string's length. A *similarity* index, $S(s_1, s_2)$, measures the degree to which a pair of objects are alike; the larger its value, the closer or more alike we think the patterns are. Conversely, *dissimilarity* coefficients, $d(s_1, s_2)$, assess the degree to which patterns differ, smaller values meaning closer or higher resemblance. Distances, differences, reciprocal of similarities, all constitute examples of dissimilarity measures. Globally, similarity and dissimilarity are referred to as *proximity* measures – $Prox(s_1, s_2)$.

Proximity values are positive numbers, its range being either bounded, such as the interval $[0; 1]$, or right unbounded: $Prox(s_1, s_2) \in [0; +\infty]$. We shall refer to proximity coefficients that have a $[0; 1]$ range as *normalized*. Given the inverse relationship between similarity and dissimilarity, a simple way to transform a similarity into a dissimilarity, in the situation of bounded ranges, is:

$$d(s_1, s_2) = \textit{maximum_similarity} - S(s_1, s_2) \quad (1)$$

2.2 Metrics and Dissimilarity Properties

The concept of proximity presented previously is very broad, imposing no restrictions to its definition other than positivity and monotonicity associated with its semantic interpretation. However, in order to either comply with perceptual intuition or to observe more elaborated mathematical formalisms, restrictions have been imposed on its definition, in the form of a set of desirable properties.

Most of the existing theories share a common important concept, namely that of geometric distance. Objects are seen as points in a metric space, $d(s_1, s_2)$ being the distance function of this space.

A *distance* or *metric*, d , is a real-valued function of two points that obeys the following properties:

- *Positivity*: $d(s_1, s_2) > 0$ and $d(s_1, s_2) = 0 \Leftrightarrow s_1 = s_2$.
- *Symmetry property*: $d(s_1, s_2) = d(s_2, s_1)$.
- *Triangle inequality*: $d(s_1, s_3) \leq d(s_1, s_2) + d(s_2, s_3)$.

The positivity property subsumes the following two distance axioms:

- *Consistency of self similarity*: $d(s_1, s_1) = d(s_2, s_2)$.
- *Minimality of self-similarity*: $d(s_1, s_1) \leq d(s_1, s_2)$.

Of the above properties, the first is generally accepted. Symmetry is also usually considered a desirable property. Concerning the triangle inequality, it is an understandable restriction under the geometrical interpretation of dissimilarity, having been explored, for instance, in the design of search techniques in the context of image retrieval [13, 6]. It basically states that, given a set of three points, the distance between any two points is at the most given by the sum of the distances between each of them and the remaining one, which occurs in the situation of colinearity.

Psychologists tend to distinguish between *perceived similarity*, expressed by d , and *judged similarity*, δ , the latter being expressed as a suitable monotonically non-decreasing function of the former: $\delta(s_1, s_2) = g[d(s_1, s_2)]$. Experimental investigation over the judged similarity, however, led to the refutation of all three properties (and underlying axioms) (see [59] and the references therein). In spite of this, metric models have been widely used in psychology and engineering application areas.

3 String Patterns and Proximity Measures

String descriptions may arise as the natural form for pattern representation, such as in the context of text documents, coding theory, and molecular biology applications, or as a result of some pattern pre-processing. Examples of the second include speech processing applications, processing of handwriting material, and image processing.

A key feature when performing pattern analysis under this representation paradigm, is the definition of a proximity measure between patterns. This is useful for analysis (comparison of sequences), classification, and clustering purposes. In the remainder of this section we present and discuss some of the proximity measures found in the literature.

3.1 Measuring Similarity between Sequences

In a wide sense, a similarity index measures the degree to which a pair of objects are alike. Concerning structural patterns represented as strings or sequences of symbols, the concept of pattern resemblance has typically been viewed from three main perspectives:

- similarity as matching, according to which patterns are seen as different viewpoints, possible instantiations or noisy versions of the same object;
- structural resemblance, based on the similarity of their composition rules and primitives;
- content-based similarity.

The string matching approach [3, 4] falls into the first category and forms the basis of most of the similarity and dissimilarity indices between sequences of symbols reported in the literature [58]. Assuming the classical string edit operations – insertion, substitution and deletion of symbols [56] – or, additionally, transposition errors [51], to which costs are associated, dissimilarity indices are typically variations of the Levensthein distance, of which the probabilistic modelling is a particular instance. Examples of dissimilarity measures related to the Levensthein distance are the Hamming distance, counting the number of mismatches over strings of the same length, and dissimilarity based on the search of longest common substrings [7]. These measures have been applied, for instance, in the comparison of sequences [41] in error correction of noisy sentences [36, 52, 51] and in recognition tasks [46, 8, 9].

Structural dissimilarity (similarity) measures the extent to which two sequences differ (are alike) in terms of rules of composition of the string patterns. One such measure is presented in [18], where the underlying structure of strings is assessed as inter-symbol dependencies, identified by means of Solomonoff's coding [62, 19]; similarity, viewed as shared subsequences, is computed as a ratio of decrease in code length. In the work reported in [16] syntactic rules model patterns' structure, and grammar complexity measures the compactness of this representation. Similarity is then defined as the ratio of decrease in grammar complexity. Fu [23] defined a distance between strings based on the modelling of a string structure by means of a grammar and on the concept of error-correcting parsing (ECP) [42, 64]. According to this model, the distance between a string and a reference string is given by the error-correcting parser as the weighted Levensthein distance between the string and the nearest (in terms of edit operations) string generated by the grammar inferred from the reference string (thus exhibiting a similar structure). This method provides a link between the structural resemblance and the string matching approaches.

In some applications, textual information comparison underly a semantic interpretation, the concept of matching being replaced by content similarity evaluation. In this case, features are extracted from the text data (such as token words or sub-strings), and conventional set-theoretical or vector-based similarity measures are applied to the resulting descriptions. The n-gram method [15] and frequency-based similarity [30, 28], constitute examples of this approach, particularly explored in the fields of text documents analysis and information retrieval. For a discussion of similarity measures and weighting functions in the context of text documents ranking see, for instance, [28] and the references therein. In this chapter we are focusing on the distinction between string matching and structural resemblance. Content-based similarity measures will not be further addressed.

In order to produce a data partition, clustering algorithms often require the definition of a similarity measure between sets of patterns (clusters). Either pairwise comparisons are performed or cluster representatives or models are adopted. According to the first approach, resemblance between pairs of patterns in distinct clusters are computed, the final similarity/dissimilarity measure being a function of these. An example in this category is the nearest-neighbor rule used in the single-link algorithm, dissimilarity between two clusters being the distance between nearest neighbor patterns. Sets of similar sequences can also be represented by a single prototype, chosen within the set (taken as the cluster center [43, 24]) or may represent the average string, such as the generalized median, defined as

No.	String
s1	ABILIO DOS SANTOS
s2	ABILIO DA CONCAICAO ALVES DOS SANTOS
s3	ABEL SANTOS
s4	ABEL LOPES SANTOS
s5	ADRIANO EDSON A. FIUSA
s6	ADRIANO EDSON FIUZA
s7	ADELAIDE CONCEICAO RODRIGUES
s8	ADELAIDE DA CONCEICAO HENRIQUES
s9	ADELAIDE CONCEICAO NATARIO FERREIRA
s10	ADELAIDE FERREIRA
s11	MARIA ADELAIDE CONCEICAO NATARIO FERREIRA
s12	ANA MARIA FARINHO MOURAO
s13	ANA MARIA FARINO MOURAO
s14	ANA PATRICIA NUNES GOMES
s15	CLARA GOMES
s16	PAULO ANTONIO GOMES

Table 1: Dataset $D1$: sample names from a customer database.

the string that has the smallest sum of distances to all strings in the set [39, 45, 34]. In this situation, the proximity between clusters is given by the proximity between their prototypes. A set of strings can also be represented by a model, such as a grammar [25, 48], a Markov chain [60] or a Hidden Markov Model (HMM) [55, 61]. Similarity between clusters are hence computed based on their model representations, being tightly related with the underlying clustering strategy.

3.2 String Alignment and Structural Similarity: Examples

The concepts of string matching and of structural similarity are illustrated in the following examples, which will be used throughout the chapter. While the first example is clearly a string matching problem, examples in sections 3.2.2 to 3.2.4 require the use of structural information, expressed either as rules of composition or as frequently occurring sub-sequences. In the example in section 3.2.5 it is not clear which type of similarity is best suited for discriminating between contour classes. Post evaluation of the different measures in the context of clustering is needed in order to make a decision, which will be addressed in section 4.

3.2.1 Names Data Set

Consider the set of strings given in table 1, representing names in a customers database. Suppose we are given the task of identifying multiple en-

tries for the same individual in the database. Repeated entries might have occurred for instance due to incorrect typing on a keyboard, misspelling of words, suppression of middle names, or recognition failure from a scanning device. Given the type of data involved, it is obvious that the order of words in the sequences is important, and alignment of sequences is needed in order to detect misspelling. Furthermore, suppression of names may be modelled by deletion of characters. The string matching approach therefore seems to be the right approach to deal with this type of patterns. Additional information about the data entry process may provide useful hints to quantify the type of errors more likely to occur, and characterize costs on string editing operations.

3.2.2 Fractal Images

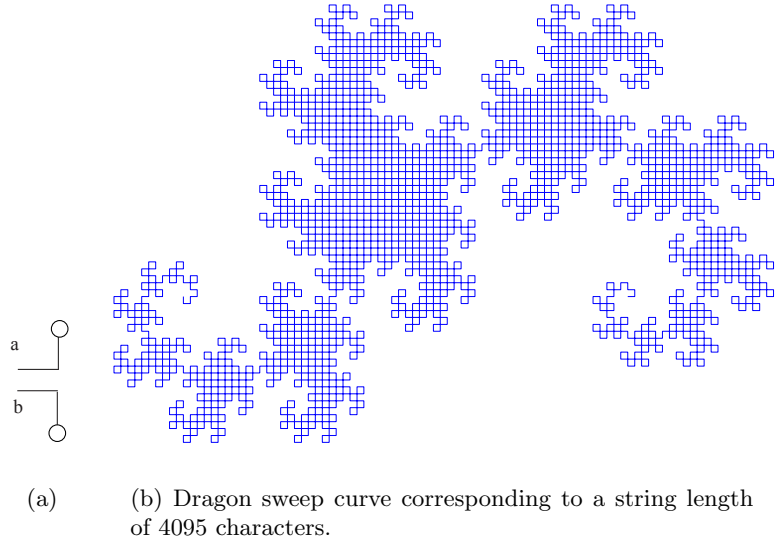
Fractals can be used as a mathematical model of many biological objects which are self-similar within a certain interval of scales. Many fractal objects result from iteration processes, which are a natural way to describe growth processes in biology and physics [35]. In a growth process the last growth stage serves as input for the next growth step. A well known model for biological pattern formation, from botany, is the Lindenmayer grammar or L-system. It is similar to grammars in conventional formal language theory, as in Chomsky hierarchy [29], the main difference being that rewriting rules are applied simultaneously in the Lindenmayer grammar, strings being generated in an iteration process.

The L-system

$$\begin{aligned}
 L_{dragon_sweep} &= (V, R, start_symbol) \\
 V &= \{a, b, +, -\}, \quad start_symbol = a \\
 R : \quad &a \rightarrow a + b, \quad b \rightarrow a - b, \quad + \rightarrow +, \quad - \rightarrow - \quad (2)
 \end{aligned}$$

condenses the morphological structure of the Dragon-Sweep fractal curve [35] shown in figure 1(b). The strings in figure 1(c) correspond to the first 5 iterations using this L-system; fractal curves are obtained from strings according to the drawing rules in fig. 1(a).

The data set *D2* consists of the first 10 iterations with this L-system, the corresponding string lengths being given in table 2. Although the string descriptions have very dissimilar lengths, all these patterns share the same rules of composition, i.e. that are structurally similar. It would be desirable to have a similarity measure that recognizes this structural resemblance, which is not addressable by string alignment techniques. Ideally, one would like to identify all these structures as being the same.



a+b
a+b+a-b
a+b+a-b+a+b-a-b
a+b+a-b+a+b-a-b+a+b+a-b-a+b-a-b
a+b+a-b+a+b-a-b+a+b+a-b-a+b-a-b+a+b+a-b+a+b-a-b-a+b+a-b-a-b

(c) First 5 strings in the iteration process.

Figure 1: Dragon sweep fractal curves. The construction of the curves, as in (b), from the string descriptions is as follows. The symbols a and b correspond to the graphical elements in figure (a); at the end point of each element (indicated as “o”), turns are made as indicated in the string. The symbol $+$ indicates a turn of 90° to the right, and $-$ is a turn of 90° to the left.

s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
3	7	15	31	63	127	255	511	1023	2047

Table 2: Data set $D2$: lengths of strings corresponding to the first 10 iterations of the L-system (2), that describes the Dragon sweep fractal.

3.2.3 Recursive Structure Line Patterns

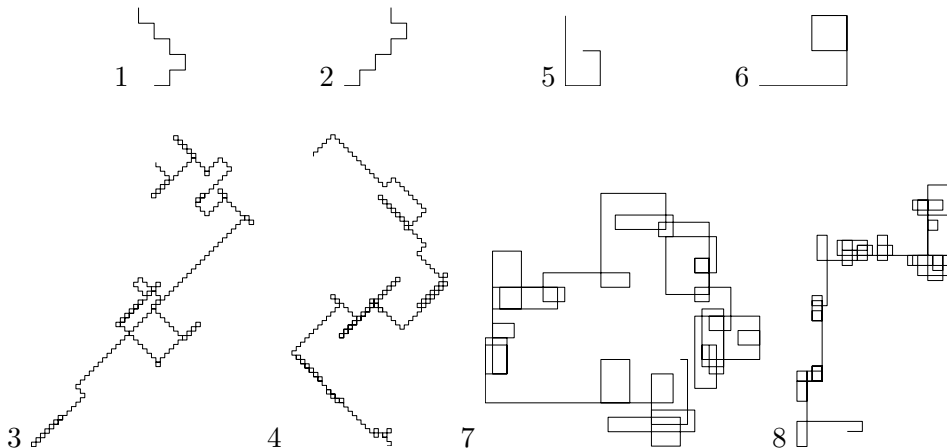


Figure 2: Data set $D3$. Graphical representation of strings of the form $(2*6)^*$ (patterns 1 to 4) and $(0*6)^*$ (patterns 5 to 8). The graphical interpretations of the symbols are as follows: 0 – maintain the line direction; 2 – turn right; 6 – turn left. The string’s lengths for the several patterns are: (1)-9; (2)-9; (3)-389; (4)-409 (5)-18; (6)-9; (7)-487; (8)-411.

Another example of string patterns with underlying grammatical rules of composition is shown in figure 2, which represents a graphical description of instances, with various lengths, of strings of the form $(2*6)^*$ or $(0*6)^*$, with the symbol $*$ indicating an arbitrary repetition of the element on the left (parenthesis are used for delimiting elements with more than one character).

3.2.4 Email Messages

The next example consists of the first lines of text extracted from email messages, as shown in table 3, by removing space characters. Variable length samples using the English and the Portuguese languages were included. The purpose is the classification of content, in the presence of variable sized sequences. While associations of words may have a fundamental role in defining topics, the order of appearance of these sequences in the text is not relevant to characterize content. The string alignment concept is meaningless in this situation. On the other hand, no general rule of string composition can be identified that could be related to content. Key-word

ID	Mail Type	Cl.	Lang.	Size	Sample Text
1	call for papers	1	Eng.	142	ISSS'98 FINAL CALL FOR PAPERS (SUBMISSION ...
2	call for papers	1	Eng.	106	CALL FOR PAPERS WORLD MULTICONFERENCE ...
3	call for papers	1	Eng.	160	CALL FOR PAPERS RENSSLAER'S INTERNAT ...
4	call for papers	1	Eng.	87	CALL FOR PAPERS THE PRAGUE STRINGOLOGY ...
5	personal letter	2	Eng.	161	DEAR ANA, THANKS FOR THE EMAIL. PLEASE ...
6	personal letter	2	Eng.	129	DEAR ANA, IN THE BACKUP DIRECTORY THERE ...
7	journal advert.	4	Eng.	137	CONTENTS DIRECT FROM ELSEVIER SCIENCE ...
8	MSc advert.	3	Port.	153	PROVAS DE MESTRADO EM ENGENHARIA ...
9	MSc advert.	3	Port.	280	PROVAS DE MESTRADO EM ENGENHARIA ...
10	MSc advert.	3	Port.	168	PROVAS DE MESTRADO EM ENGENHARIA ...

Table 3: Data set *D4*. Text from email messages.

based or sub-sequence based string comparisons seem more valuable tools for this problem.

3.2.5 Contour Images

The last data set, *D5*, used consists of string descriptions of contour images from three types of hardware tools, as follows (see table 4): 10 samples from tool type 1; tool type 2 has two poses (closed and half opened), 10 samples per pose being used; tool type 3 also has two poses (wide opened and half opened), 10 samples per pose. Each image was segmented to separate the object from the background, and the object boundary was sampled at 50 equally spaced points [22]. The strings correspond to encoding objects contours using an 8-directional differential chain code [31].




Class	Tool	Samples
1		61072620026270163702620026207261072601627026200262 71172620026200162701637026207261073610726116370262 71172620026270262002620737016370262002620026200162
2		00000000660000001710000060000010000000075000000003 00000000500000000000000760000016100000066000000003 00000000500000001710000050000016100000076000000003
3		10000000050000000007207000701700000000050000000012 1000000005000000000107000702700000000050000000012 10000000057000000007107007702700000000050000000012

Table 4: Data set *D5*. Sample string contour descriptions of hardware tools.

3.3 Dissimilarity based on String Editing Operations

A very well known measure of dissimilarity between strings is the *Levenshtein distance*. It is based on the definition of string editing operations or error transformations, namely substitution, deletion and insertion of symbols:

- $T_S(b|a) \rightarrow$ substitution of a by b , $a \neq b$
- $T_S(a|a) \rightarrow$ maintaining symbol a
- $T_I(b) \rightarrow$ insertion of b
- $T_D(a) \rightarrow$ deletion of a

The Levenshtein distance between two strings $s_1, s_2 \in \Sigma^*$, $D_L(s_1, s_2)$, is defined as the minimum number of editing operations needed in order to transform s_1 into s_2 . An extension of the Levenshtein distance by associating differentiated costs to the several editing operations is known as the *weighted Levenshtein distance*, defined as

$$d_W(s_1, s_2) = \min \{ \gamma(T) | T \text{ is an edit transformation of } s_1 \text{ into } s_2 \} \quad (3)$$

where $\gamma(T) = \sum_{i=1}^m \gamma(T_i)$, T_i is the i th editing operation performed in the transformation T , and $\gamma(T_i)$ is the cost associated with that operation. Figure 3 illustrates the matching of two strings according to editing operations. A particular instance of this weighting function is the probabilistic modelling, according to which the following normalization condition must hold:

$$\sum_{b \in \Sigma} \gamma(T_S(b|a)) + \sum_{b \in \Sigma} \gamma(T_I(b)) + \gamma(T_D(a)) = 1. \quad (4)$$

In order to preserve the symmetry property, weights can not be assigned arbitrarily. The costs of inserting or deleting a given symbol must be equal, as for the substitution of symbols: $\gamma(T_S(b|a)) = \gamma(T_S(a|b))$. It can be shown that the triangle inequality is verified and thus it constitutes a metric (see for instance [8] and the references therein).

Normalization of the previous distances with respect to string lengths are obtained by post-normalization (normalized weighted Levenshtein distance)

$$d_{NW}(s_1, s_2) = d_W(s_1, s_2) / \max\{|s_1|, |s_2|\}, \quad (5)$$

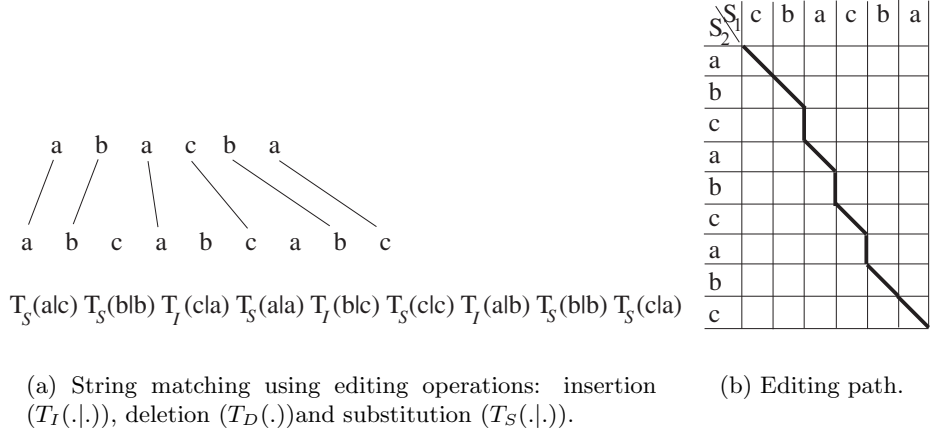


Figure 3: String matching. In (b) diagonal path segments represent substitutions, vertical segments correspond to insertions, and horizontal segments correspond to deletions.

or by optimal minimization of the distance normalized by the length of the editing path – normalized edit distance [46]

$$d_{NED}(s_1, s_2) = \min \left\{ \frac{\gamma(T)}{|T|} \mid T \text{ is an editing path between } s_1 \text{ and } s_2 \right\}, \quad (6)$$

where $|T|$ is the length of the editing path (see figure 3(b)). Vidal [46] has shown that the latter performs better in many situations. It should be emphasized that normalization leads to dissimilarity measures that are not metrics due to failure of the triangle inequality [8].

	s2	s3	s4	s5	s6	s7	s8	s9	s10
d_W	1.6	1.5	4	8.5	9.4	8.2	8.3	7.8	10.6
$d_{NED} (\times 1e - 1)$.52	1	2.3	3.4	3.9	2.7	2.6	2.2	4.3

Table 5: Distances between string $s1$ and strings $s2$ to $s10$ in data set $D1$.

Table 5 presents weighted Levensthein distances and normalized edit distances between string $s1$ and strings $s2$ to $s10$ in data set $D1$. In order to model omission of middle names, the deletion operation was given a low cost (0.1) in comparison to the other error transformations (cost=1). As shown, without normalization, string $s1$ (ABILIO DOS SANTOS) is consid-

ered more similar to the third string (ABEL SANTOS) – $d_W = 1.5$, than to the second string (ABILIO DA CONCAICAO ALVES DOS SANTOS) – $d_W = 1.6$. By using normalization, however, the reverse situation occurs, with a more clear distinction between the two dissimilarities ($d_{NED}(s1, s2) = 0.052$ and $d_{NED}(s1, s3) = 0.1$), which is in better agreement with perceptual expectations.

For simplicity, and under the assumption of no *a priori* knowledge about the penalizing mechanisms of the error transformations, in the remaining of the chapter all edit operations will be assigned unitary costs, except for the operation of maintenance of a symbol, which will be assigned a null weight.

	s2	s3	s4	s5	s6	s7	s8	s9	s10
d_W	4	12	28	60	124	252	508	1020	2044
$d_{NED} (\times 1e - 1)$	5.71	8.00	9.03	9.52	9.76	9.88	9.94	9.97	9.98

Table 6: Distances between string $s1$ and strings $s2$ to $s10$ in data set $D2$.

The dependency of these dissimilarity measures on the string lengths (even the normalized distances) is illustrated in table 6, with the fractal curve data set ($D2$).

3.4 Dissimilarity based on Error-Correcting Parsing

Fu [24, 23] defined a distance between strings based on the modelling of string structure by means of grammars and on the concept of error-correcting parsing [42, 64]. According to this model, the distance between a string and a reference string is given by the error-correcting parser as the weighted Levensthein distance between the string and the nearest (in terms of edit operations) string generated by the grammar inferred from the reference string (thus exhibiting a similar structure):

$$d_{ECP_f}(s_1, s_2) = \min_y \{d_W(s_1, y) | y \in L(G_{s_2})\} \quad (7)$$

$G_{s_2} := \text{Grammar inferred from } s_2\}$

The computation of the dissimilarity between strings requires two steps: (1)- modelling of strings with grammars; (2)- computation of the dissimilarity by error-correcting parsing. The process is schematically described in figure 4. The grammatical inference procedure is responsible for the identification of regular patterns of composition that lead to the definition of rules. Different inference algorithms will produce distinct results both quantitatively and qualitatively, depending on how far apart are the underlying

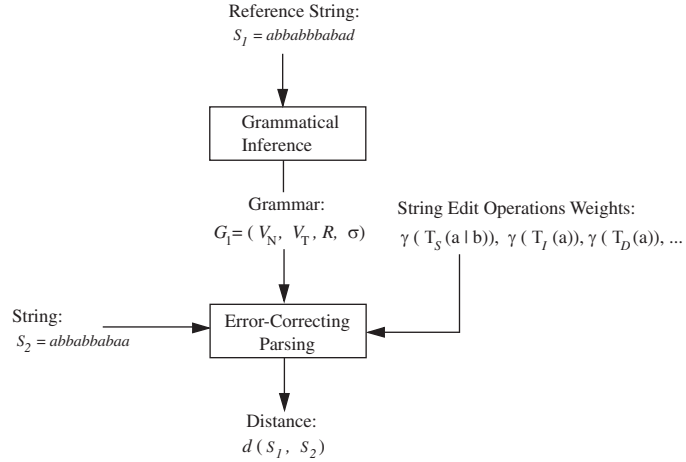


Figure 4: Computation of dissimilarity between strings using error-correcting parsing.

heuristics or criteria supporting the methods. It is not possible to define the “optimal” grammatical inference algorithm. Several methods can be tested or the election of a specific algorithm has to be based on some assumption or prior knowledge about the underlying structure. Crespi-Reghizzi’s method [25, 48] will be used for grammatical inference in this chapter.

It is important to notice that, even with adequate definition of the editing weights, the symmetry property cannot be ensured *a priori*. In fact, $d_{ECP_f}(s_1, s_2) = \min\{d_L(s_1, L(G_{s_2}))\}$, which, in general, will be different from $d_{ECP_f}(s_2, s_1) = \min\{d_L(s_2, L(G_{s_1}))\}$. For instance, given the strings $s_1 = a + b$ and $s_2 = a + b + a - b$, from the data set D_2 , $d_{ECP_f}(s_1, s_2) = 1$ (a recursive structure is identified by the inference algorithm on the second string) while $d_{ECP_f}(s_2, s_1) = 4$. In order to preserve the symmetry property, we will use a new definition:

$$d_{ECP}(s_1, s_2) = \min\{d_{ECP_f}(s_1, s_2), d_{ECP_f}(s_2, s_1)\} \quad (8)$$

We shall designate by d_{NECP} the previous dissimilarity measure normalized by the edit path. Neither d_{ECP} or d_{NECP} constitute a metric because they do not obey the triangle inequality (for instance in the data set D_1 , $d_{ECP}(s_2, s_1) = 5$, $d_{ECP}(s_1, s_6) = 11$ and $d_{ECP}(s_2, s_6) = 23$).

The capacity of these dissimilarity measures to capture structural resemblance, by means of inferred grammars, is shown in table 7, concerning

	$(s1, sj), j > 1$	$(s2, sj), j > 2$	$(si, sj), i, j > 2, i \neq j$
d_{ECP}	1	1	0
d_{NECP}	.33	.14	0

Table 7: Distances between pairs of strings in data set $D2$ using error-correcting parsing.

the comparison of strings in the data set $D2$. Zero distance is obtained between pairs of strings produced after 3 or more steps of the L-system iteration process. The non-zero values for strings $s1$ and $s2$ result from the fact that these strings do not have the $-a$ or $-b$ structures, which are present in all remaining sequences, and therefore are included in the corresponding inferred grammatical models.

3.5 Grammar Complexity-based Dissimilarity

Grammars are compact representations of structural rules of composition in string patterns. Reference [16] assumes this formalism for modelling string patterns, exploring the concept of grammar complexity in the definition of a new measure of similarity between strings. The basic idea is that, if two sentences are structurally similar, then their joint description will be more compact than their isolated description due to sharing of rules of symbol composition; the compactness of the representation is quantified by the grammar complexity, and the similarity is measured by the *ratio of decrease in grammar complexity*, as follows:

$$RDGC(s_1, s_2) = \frac{C(G_{s_1}) + C(G_{s_2}) - C(G_{s_1, s_2})}{\min \{C(G_{s_1}), C(G_{s_2})\}}, \quad (9)$$

where $C(G_{s_i})$ denotes grammar complexity.

Let $G = (V_N, \Sigma, R, \sigma)$ be a context-free grammar, where V_N, Σ are the sets of nonterminal and terminal symbols, respectively, σ is the grammar start symbol and R is the set of productions written in the form:

$$\begin{aligned} A_1 &\rightarrow \alpha_{11} \mid \dots \mid \alpha_{1l_1} \\ &\vdots \\ A_r &\rightarrow \alpha_{r1} \mid \dots \mid \alpha_{rl_r} \end{aligned}$$

Let $\alpha \in (V_N \cup \Sigma)^*$, be a grammatical sentence of length n , in which the symbols a_1, a_2, \dots, a_m appear k_1, k_2, \dots, k_m times, respectively. The complexity

of the sentence, $C(\alpha)$, is given by [25, 48]

$$C(\alpha) = (n + 1)\log(n + 1) - \sum_{i=1}^m k_i \log k_i. \quad (10)$$

The complexity of the grammar G , $C(G)$, is defined as [16]

$$C(G) = \sum_{i=1}^r \sum_{j=1}^l C(\alpha_{ij}) \quad (11)$$

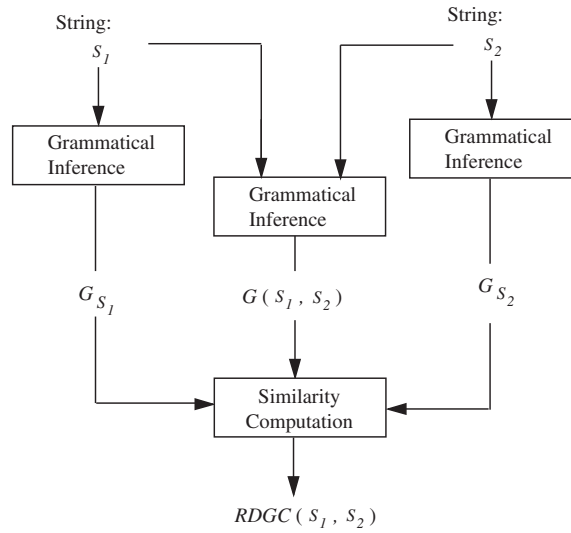


Figure 5: Computation of similarity between strings based on grammar complexity.

The computation of this similarity, as schematized in figure 5, is basically supported on grammatical inference and simple evaluation of the resulting rewriting rules. Computational complexity is therefore conditioned by the inference algorithm. When using Crespi-Reghezzi's method, strings are processed individually; the inference of a grammar for a pair of strings takes advantage of the previous computations, consisting of a simple merging of individually inferred grammars.

Similarity values obtained with this method, as well as for the preceding method, depend on the grammatical inference strategies adopted. However, the following characteristics hold, independently of the grammatical infer-

ence algorithm:

$$\begin{aligned}
RDGC(s_1, s_2) &\geq 0 \\
&\leq 1 \\
&= 1 \text{ if } s_1 \equiv s_2 \\
&= 0 \text{ if } s_1 \text{ and } s_2 \text{ have non overlapping alphabets} \\
RDGC(s_1, s_2) &= RDGC(s_2, s_1)
\end{aligned}$$

As indicated above, the symmetry property is verified; the triangle inequality, however, is not always preserved.

S_{RDGC}	$(s1, sj), j > 1$ 0.15	$(s2, sj), j > 2$ 0.85	$(si, sj), i, j > 2, i \neq j$ 1
------------	---------------------------	---------------------------	-------------------------------------

Table 8: Minimum grammar complexity-based similarity between pairs of strings in data set $D2$.

The similarity values between pairs of strings in the data set $D2$ is summarized in table 8. As seen before, strings $s1$ and $s2$ do not contain enough variants of the L-system rules of composition, the inferred grammars not being able to generalize to the recognition of the remaining structures; on the other hand, grammars inferred from the longer strings also do not recognize $s1$ and $s2$. As a result, the similarity between strings involving $s1$ or $s2$ is low, identification of a unique structure being achieved for longer strings (similarity=1).

3.6 Minimum Code Length-based Similarity

A measure of structural similarity exploring the notion of compressibility of sequences and algorithmic complexity, is proposed in [18]. A string is said to be compressible if there exists a description shorter than the original sequence. Low compressibility is related with high complexity and randomness; on the other hand, sequences with inter-symbol dependencies, showing subpattern regularities, are likely to have compact descriptions. Solomonoff's code [62] is then used for the search of pattern regularities and sequence compression. According to this coding scheme, a sequence s_i is represented by the triplet:

$$alphabet_description, special_symbols_definition, coded_string,$$

where a coded string is obtained in an iterative procedure where, in each step, intermediate codes are produced by defining sequences of two symbols,

which are represented by special symbols, and rewriting the sequences using them. Compact codes are produced when sequences exhibit local or distant inter-symbol interactions.

This coding scheme has been extended to sets of strings

$$alphabet, symbol_definition, s_1_coded, s_2_coded, \dots, s_m_coded$$

and global compact codes are produced by considering the inter-symbol dependencies on the ensemble of the strings. Strings sharing subpattern regularities will therefore produce more compact codes than the gathering of the codes for the individual sequences. The quantification of this reduction in code length forms the basis of the similarity measure which will be designated by NRDCCL. The length of the codes produced using Solomonoff's method (given by the sum of the lengths of the descriptions of the three part code above) is evaluated for the isolated strings and for the ensemble; NRDCCL is defined as the normalized ratio of decrease in code length obtained by the association:

$$NRDCCL(s_1, s_2) = \frac{code_len(s_1) + code_len(s_2) - code_len(s_1, s_2)}{(as)/2 + abs(|s_1_coded| - |s_2_coded|)} \quad (12)$$

with

$$as = |alphabet_{s_1}| + |symbol_def_{s_1}| + |alphabet_{s_2}| + |symbol_def_{s_2}|$$

The order of appearance of sub-sequences is irrelevant for the computation of code lengths. This similarity measure therefore totally departs from string matching paradigms, structure being related to inter-symbol dependencies and not rules of string composition. Accounting for highly frequent sub-sequences, Solomonoff coding assumes long strings. Results with short strings may therefore be meaningless. This is illustrated in table 9, where similarities involving strings shorter than or equal to 31 characters (string s4) are very low.

	s1	s2	s3	s4	s5	s6	s7	s8	s9	s10
s1	1	0.4	0.19	0.10	0.05	0.19	0.14	0.12	0.09	0.08
s10	0.08	0.19	0.39	0.61	0.74	0.85	0.92	0.97	1	1

Table 9: Minimum code length based similarity between strings in data set *D2*.

4 Clustering of String Patterns

The previous proximity measures are evaluated in this section in the context of clustering. Clustering results are strongly conditioned by the underlying similarity measures, but also by the intrinsic cluster validity criteria and structure imposed on the data. Several clustering algorithms integrating the previous proximity measures will therefore be investigated. Table 10 characterizes the clustering algorithms tested. They are further detailed in the next section.

4.1 Clustering Algorithms Tested

a) Hierarchical Agglomerative Schemes			
Philosophy of the method	Proximity between clusters	String sim.	Acronym
String edit operations	Nearest-neighbor rule: $d(C_i, C_j) = \min \{d(s_k, s_l) s_k \in C_i, s_l \in C_j\}$	d_{NED}	SL-NED
Grammar complexity	Ratio of decrease in grammar complexity: $RDGC(C_i, C_j) = \frac{C(G_{C_i}) + C(G_{C_j}) - C(G_{C_i, C_j})}{\min\{C(G_{C_i}), C(G_{C_j})\}}$	$RDGC$	MGC
Minimum code length	Normalized decrease in code length: $NRDCL(C_i, C_j) = \frac{code_len(C_i) + code_len(C_j) - code_len(C_i, C_j)}{(as)/2 + abs(C_i_coded - C_j_coded)}$ $as = \sum_{k=i,j} alph_C_k + symb_def_C_k $	$NRDCL$	MCL
b) Sentence to Sentence Clustering			
Error correcting parsing	Nearest-neighbor rule: $d_{NECP}(s_i, C_j) = \min\{d(s_i, y) y \in L(G_{C_j}, \cdot)\}$	d_{NECP}	ECP-NED

Table 10: Characterization of clustering algorithms under study.

4.1.1 Hierarchical Agglomerative Clustering

The hierarchical agglomerative clustering method can be schematically described as follows:

Input: A set of n strings $D = \{s_1, s_2, \dots, s_n\}$; a threshold th or the desired number of clusters N_c .

Output: A partition of D into m clusters ($m = N_c$ if the latter is specified), C_1, C_2, \dots, C_m .

Steps:

1. Set each pattern into a single cluster. Set $m = n$.
2. Compute the proximity matrix between all pairs of strings.
3. If $m = N_c$ go to step 6.
4. Select the most similar pair of clusters, C_i, C_j , from the proximity matrix and let *simil* be the value of this similarity. If *simil* < *th* (*simil* > *th*, if it is a dissimilarity measure), go to step 6.
5. Merge the two clusters and update the proximity matrix accordingly. Set $m = m - 1$ and continue in step 3.
6. End the computation by returning the data partition encountered.

Different clustering algorithms are obtained from the previous method by defining the proximity measure between patterns and between clusters. The first algorithm in table 5 is based on string editing operations, and corresponds to using the normalized edit distance (see equation 6 in section 3.3) as dissimilarity measure between strings (step 2), and adopting the nearest-neighbor rule (single-link method) for defining distances between clusters (step 5):

$$d(C_i, C_j) = \min \{d_{NED}(s_k, s_l) \mid s_k \in C_i, s_l \in C_j\}. \quad (13)$$

The second algorithm explores the minimum grammar complexity concept (section 3.5). Similarity between strings is computed using equation 9; this is extended to sets of strings, as given by the equation

$$RDGC(C_i, C_j) = \frac{C(G_{C_i}) + C(G_{C_j}) - C(G_{C_i, C_j})}{\min \{C(G_{C_i}), C(G_{C_j})\}}, \quad (14)$$

where G_{C_i} is the grammar inferred from strings in cluster C_i . Additionally to a data partition and the dendrogram expressing the relationships between string patterns, this method provides a model for each cluster: the corresponding grammar. Results reported in this chapter concern the application of Crespi-Reghezzi's method for grammatical inference. When dealing with long strings, the grammatical inference process is computationally more efficient (it is linear on strings length) than the computation of the normalized edit distance, and therefore this clustering method becomes faster than the first algorithm.

The last hierarchical clustering algorithm computes distances between strings by the normalized ratio of decrease in code length (see equation 12 in section 3.6). A code for the set of strings in a cluster is computed using the extension of Solomonoff’s code described in section 3.6; similarity between clusters is obtained by replacing single strings by sets of strings (representing clusters) in equation 12, leading to

$$NRDCL(C_i, C_j) = \frac{code_len(C_i) + code_len(C_j) - code_len(C_i, C_j)}{(as)/2 + abs(|C_i_coded| - |C_j_coded|)} \quad (15)$$

4.1.2 Sentence-to-Sentence Clustering

The sentence-to-sentence clustering algorithm is a partitional type method proposed by Fu [24, 23] based on the concept of error correcting parsing. Grammars are used to model cluster structure. The algorithm starts with a single cluster containing the first sample; the remaining data is classified by computing the (weighted) Levensthein distance between the candidate string and the most similar string (determined by error-correcting parsing) modelled by the grammars describing the clusters formed so far; if the minimal distance found is less than a given threshold, th , the string is included in the corresponding cluster and its grammar is updated; otherwise, a new cluster is formed with this sample.

This clustering strategy implements a nearest-neighbor rule, with sequences being compared, not directly with patterns previously included in clusters, but with the best matching elements in the languages generated by the grammars inferred from the clustered data. The implementation used in this chapter will apply Crespi-reghizzi’s algorithm for grammatical inference, without assuming *a priori* information. The dissimilarity measure based on error-correcting parsing defined in equation 7 (section 3.4) is extended for the computation of dissimilarity between a string, s_i , and a cluster, C_j , as

$$d_{ECP}(s_i, C_j) = \min\{d_W(s_i, y) \mid y \in L(G_{C_j})\}, \quad (16)$$

where G_{C_j} is the grammar inferred from the strings in cluster C_j . Without *a priori* information, 0-1 costs will be assumed for the the string editing operations (corresponding to the Levensthein distance); in order to reduce the dependency on string lengths, distances will be normalized by the length of the editing path:

$$d_{NECP}(s_i, C_j) = \min\{d_{NED}(s_i, y) \mid y \in L(G_{C_j})\}. \quad (17)$$

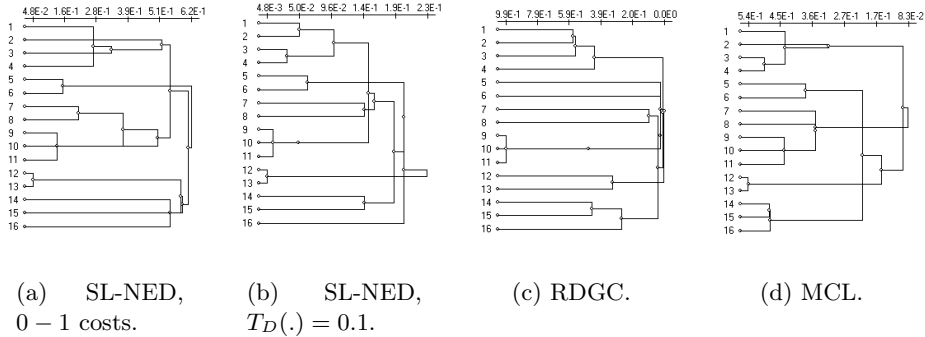
Due to the nature of this similarity measure (asymmetric) and the clustering strategy, results obtained with the sentence-to-sentence clustering algorithm may depend on initialization and on the order of pattern presentation.

Concerning computational complexity, this clustering scheme requires fewer comparisons between strings than the hierarchical one. While hierarchical methods require the computation of $n(n - 1)/2$ entries in the initial proximity matrix (the iterative process requires the update of this matrix, one line per iteration, as the size of the proximity matrix decreases), according to the sentence-to-sentence technique, each string is processed only once and compared with a, usually, reduced number of clusters. The processing of each string, however, involves grammatical inference (which can be made iterative and linear on the strings length, L) and error-correcting parsing, which in general has $O(L^2)$ time complexity. Thus, depending on the dimensions of the data sets and of the strings, one method or the other may be more advantageous from the computational complexity perspective.

4.2 Experimental Results

4.2.1 Names Data Set

Figure 6 presents the results of clustering of the names data set using several clustering algorithms. Methods based on string edit operations are strongly dependent on the costs assigned. This is illustrated in figures 6(a) and 6(b), showing the dendrograms produced by the SL-NED method for two different assignments for the deletion costs: (a)- $T_D(\cdot) = 1$; (b)- $T_D(\cdot) = 0.1$. The costs for all remaining error operations are set to 1 (no-error cost=0). Situation (b), reducing the penalty for deletion operations, gives the best pattern association results. For instance, the relations between patterns 1 to 4 in (a) are not “natural” given the context, string s_1 being found more similar to string s_4 (different first names) than to s_2 (corresponding to the inclusion of middle names). This is better modelled in the dendrogram in (b). Since the lengths of the strings are not too different and normalized edit distances are being used, the results with error correcting parsing (ECP-NED) are similar to the ones produced by the string matching approach (SL-NED), the first directly proposing a data partition by cutting the minimum spanning tree at a given level, and the second producing a dendrogram. For example, the partition obtained with the ECP-NED with $th = 0.3$ (see figure 6(d)) corresponds to cutting the dendrogram in figure 6(a) at the same level.



(1 4), (2), (3), (5 6), (7 8), (9 11), (10), (12 13), (14), (15), (16)

(e) ECP-NED, $th = .3..$

Figure 6: Clustering of the names data set. Figures (a) to (d) represent dendrograms produced by the hierarchical clustering algorithms: the indices of the patterns are indicated on the left; the merging of clusters is graphically displayed by joining lines, and the corresponding proximity values are read from the scale on the top.

The method based on minimum grammar complexity (RDGC, see figure 6(c)) gives inadequate pattern associations, significant similarity being found only between strings s_9 - s_{10} - s_{11} . The MCL technique, accounting for similar names, performs better (see figure 6(d)). The string matching paradigm, however, proves to be more appropriate for this problem, results being more in agreement with perceptual evaluation of the patterns.

4.2.2 Fractal Images

The structure of the Dragon-Sweep data set, based on grammatical rules of composition, is easily handled by the grammar-based clustering techniques, RDGC (see figure 7(b)) and ECP-NED. As seen before, both recognize maximum similarity between string patterns of length greater than 7 characters; shorter strings are not recognized as having the same structure because they do not make use of certain structures consistently identified in longer strings. Without *a priori* information, inference algorithms are not able to identify a common set of rules for all strings. As a result, both clustering algorithms

will partition the data into two clusters $((s1), (s2 \dots s10))$ or three clusters $((s1), (s2), (s3 \dots s10))$, by an appropriate setting of the design parameter th . Neither SL-NED nor MCL methods can adequately handle these patterns, as shown in figures 7(a) and 7(c), depending on strings length in a linear (SL-NED) or non-linear way.

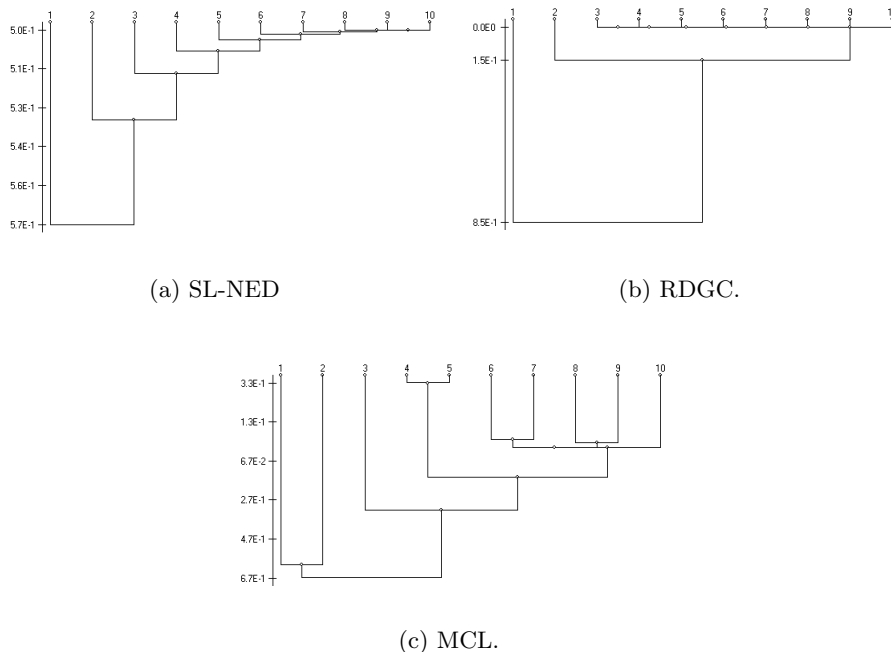


Figure 7: Clustering of the Dragon-Sweep data set.

4.2.3 Recursive Structure Line Patterns

The structural resemblance of string patterns in this example is correctly captured by the model-based techniques. The error-correcting parsing based clustering method correctly identifies two clusters in the data, for a wide range of values of th ($0.1 < th \leq 0.6$). Similar results are obtained with the minimum grammar complexity clustering algorithm, a clear separation between the two clusters being evident in the dendrogram in figure 8(b). The results obtained with the SL-NED show the dependency of the dissimilarity measure on the strings length, higher similarity being found between

matching sized strings: the dendrogram in figure 8(a) shows that strings s_1 and s_2 are first joined with strings s_5 and s_6 (with different structure but similar length); strings s_3 and s_4 will join these when a single cluster with all strings is formed. The MCL method is unable to identify the correct structure of the data (see figure 8(c)).

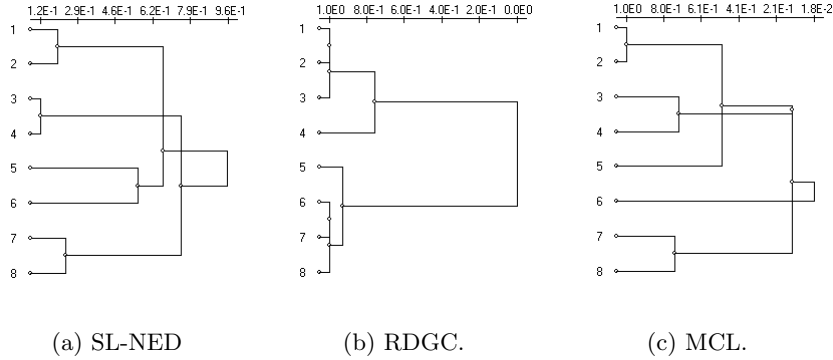


Figure 8: Clustering of the patterns in figure 2.

4.2.4 Email Messages

The categorization of email text is essentially dependent on content, the order of the narrative being arbitrary. In this case, only the MCL method was able to provide meaningful associations (figure 9(c)), correctly forming the classes 1 to 3; sample 7 (class 4) was assigned to class 1 due to the existence of a common theme in the conference and journal advertisements. Low similarity values (< 0.01) were found by the RDGC method at the character level, no significant associations being made. Methods based on error transformations failed to discover common patterns in the data (figures 9(a) and 9(b)). The normalized dissimilarity values of the SL-NED method were high and in narrow ranges, resulting in no associations for threshold values above 0.73; figure 9(a) corresponds to imposing 4 classes in the data.

4.2.5 Contour Images

Figure 10 shows the results of clustering of the contour images data set. The correspondence between string indices and tool classes is given by: s_1 to s_{10} – tool 1; s_{11} to s_{20} – tool 2, pose closed; s_{21} to s_{30} – tool 2, pose

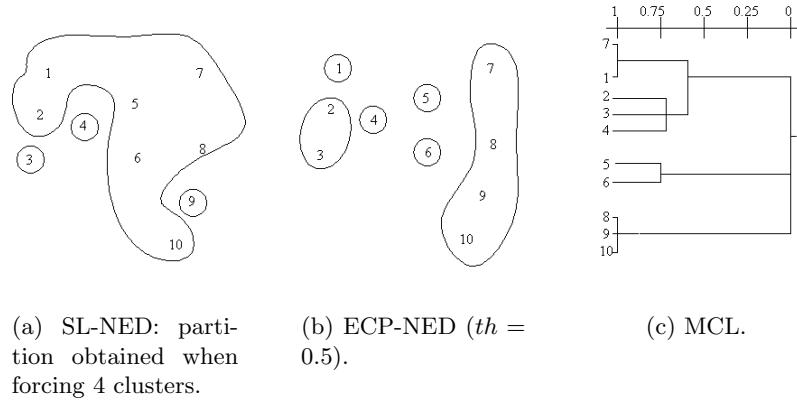


Figure 9: Clustering of the email data set.

half opened; s_{31} to s_{40} – tool 3, pose half opened; s_{41} to s_{50} – tool 3, pose wide opened. While tool 1 is quite different from the remaining, tools 2 and 3 are more similar to each other.

Analysis of the dendrograms in figure 10 show that, based on a nearest-neighbor rule, all methods tend to organize string patterns based on pose rather than on tool shape: while tool 1 is well isolated from the remaining classes in all dendrograms, opened poses (wide open and half opened) are first gathered in the dendrograms (tool 3 first joined and then connected with tool 2 - half opened, with the SL-NED method, half-opened tools 2 and 3 being merged first with the RDGC technique, and no clear separation being noticed with the MCL algorithm) and only after that the strings from tool 2, pose closed, are joined. By thresholding on the dendrograms, two classes are obtained with the SL-NED method: tool 1 vs (tool 2 + tool 3). The RDGC method clearly separates tool 1 and tool 2, pose closed; the identification of tool 3, pose wide opened, would imply a splitting of the half-opened tools into several clusters. In order to avoid single element clusters, the threshold is selected to detect two clusters with the MCL technique: tool 1 vs remaining tools.

Partitions produced with the error-correcting parsing based method (ECP-NED) are similar to the ones obtained with RDGC, except that no separation between classes is achieved without obtaining single-element clusters. Setting the threshold to 0.1, four clusters are obtained: an element from tool 1 is detected as an outlier (single-element cluster), the remaining strings

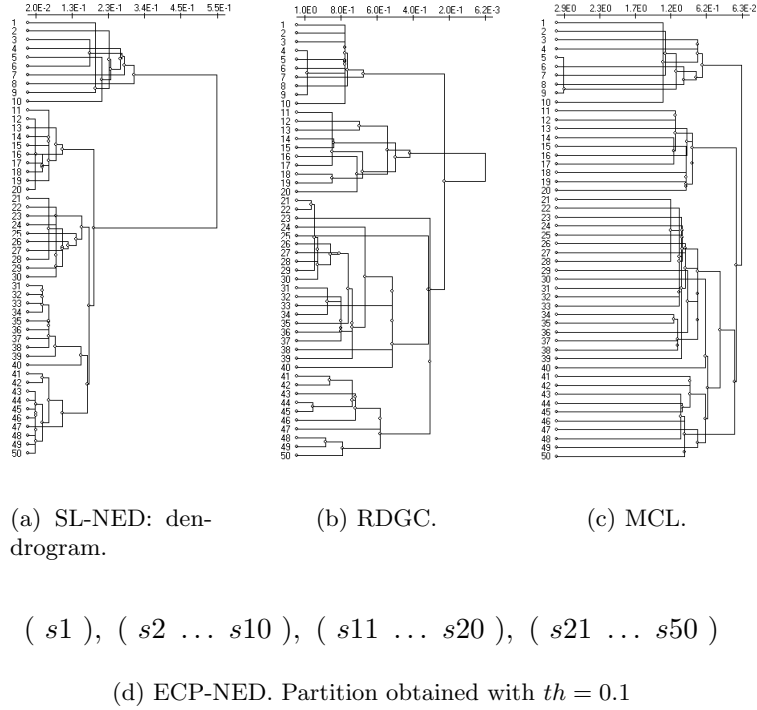


Figure 10: Clustering of the contour images data set. (a) to (c) represent dendrograms produced by the indicated hierarchical methods.

forming another cluster; tool 2, pose closed also constitutes a cluster; the fourth cluster gathers the remaining strings. Although the strings have a fixed length (50 characters) and differentiation between classes are present in the dendrograms using the string matching paradigm, the grammar complexity similarity measure leads to a better normalization of dissimilarities, making it simpler to detect classes by cutting the dendrograms.

4.3 Conclusions

Both model-based techniques, supported on grammatical models (ECP-NED and RDGC), are able to handle structural information where the order of sub-patterns in strings is important. The error-correcting parsing method combines the advantages of model based dissimilarity (being independent on strings length) with the string matching mechanism. Its major limitations

are the computational complexity, and the lack of criteria to set the design parameter, *th*. Being a hierarchical method, cluster isolation criteria [10, 1, 21] can be adopted to extract a data partition from the dendrogram produced by the RDGC method. Additionally to a data partition, both methods provide a compact model describing the identity of patterns within a cluster.

The string matching paradigm, on the contrary, cannot handle structural similarity, being sensitive to strings length. The MCL method is the only method, among the ones under study, that recognizes similarity based on inter-pattern dependencies, where the order of sub-sequences is irrelevant.

4.4 Current Trends in Clustering

In the previous sections, several similarity measures have been described and discussed, being further analyzed in the context of clustering, either exploring nearest-neighbor based hierarchical agglomerative methods, or a partitional type method using the error-correcting parsing based dissimilarity. As seen, the similarity measures between string patterns strongly condition the final partitions produced. The similarity measure between clusters is another variant that, in particular in hierarchical methods, is related to the cluster shape concept. While the RDGC and the MCL method define a similarity between clusters by extrapolating the concept of similarity between strings, the SL-NED algorithm, which is a simple single-link method by using as proximity matrix the (weighted normalized) Levenstein distance, is easily adapted to other hypothesized clusters shapes. Figure 11(a) presents the dendrogram produced on the contour images data set, when adopting the farthest neighbor rule for updating the dissimilarity matrix, under the normalized string edit distance paradigm: complete link method over the same initial dissimilarity matrix (CL-NED). As shown, patterns tend now to be organized by tool type rather than pose. Thesholding on this dendrogram will, nevertheless, produce the same two-cluster solution. Still, using the hierarchical agglomerative technique, a cluster isolation criterion that is able to handle this distinct data sparseness situation has been recently proposed [21]. The cluster isolation criterion is based on the computation of a statistic of dissimilarity increments within a cluster, isolation of a cluster being produced when the increment towards further pattern associations is high in comparison with the cluster statistic (see [21] for a more detailed description of the method). Figure 11(b) shows the dendrogram obtained when adopting this cluster isolation criterion with the complete-link method over the NED matrix, with three clusters being formed, each corresponding

to a different tool.

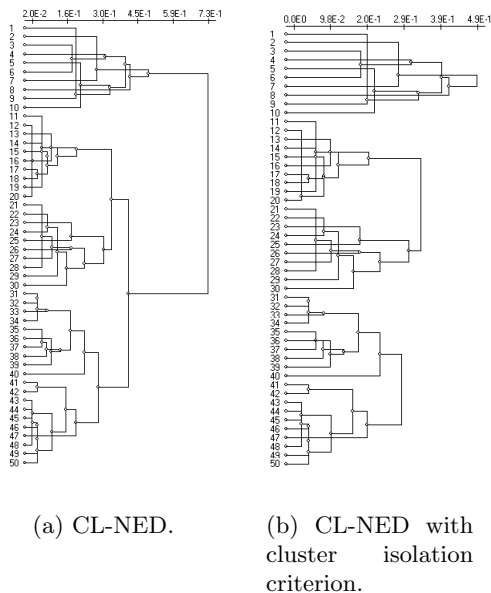
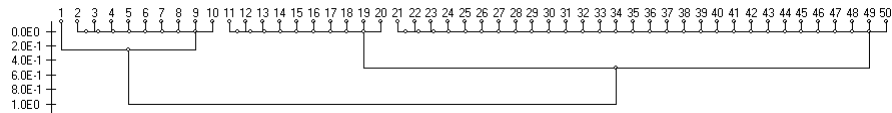


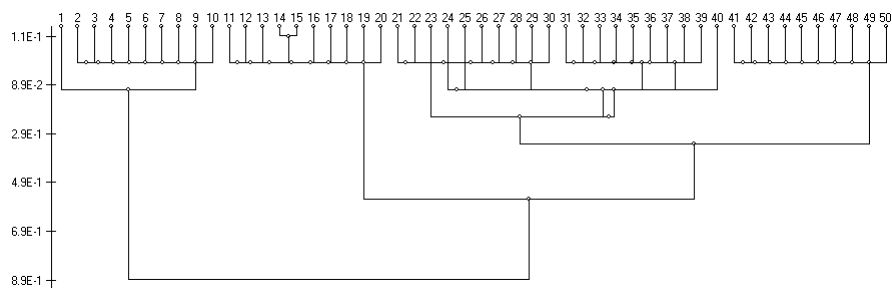
Figure 11: Clustering of the contour images data set. (a) dendrogram produced by the farthest neighbor rule applied to the normalized edit distance between pattern pairs; (b) corresponding dendrogram when adopting a cluster isolation criterion based on the analysis of dissimilarity increments statistics [21].

Other clustering algorithms based on the previous proximity measures could be developed. For instance, single link or complete link methods applied to an initial proximity matrix based on the ECP, RDGC or NRDCL paradigms; partitional type clustering, such as the K-means algorithm, using either of the proximity definitions, etc.

Different combinations of clustering strategies with dissimilarity paradigms will in general produce distinct results. The question that arises is: what is the best strategy to use? As shown in previous examples, the quality of a similarity measure/clustering method depends on the particular application problem. Knowledge about the situation modelled may dictate a particular measure as the most natural, and eventually determine underlying parameters and clustering strategies. Alternatively, several algorithms can be tested and checked for their performance.



(a) Combination of the partitions obtained in section 4.2.5 (four clusterings) using a voting mechanism.



(b) Including more clustering results, exploring, for instance, the farthest neighbor rule in hierarchical clustering.

Figure 12: Combining clustering results.

Inspired by the work in sensor fusion and classifier combination techniques in pattern recognition [37, 38], the work reported in [17] proposes a combination of clusterings in order to devise a consistent data partition. The idea of evidence accumulation clustering is to combine the results of multiple clusterings into a single data partition, by viewing each clustering result as an independent evidence of data organization. In [17] the K-means algorithm is used as the basic algorithm for decomposing the data into a large number, k , of compact clusters; evidence on pattern association is accumulated, by a voting mechanism, over multiple clusterings obtained by random initialization of the K-means algorithm. This produces a mapping of the clusterings into a new similarity measure between patterns, summarized in a matrix, co_assoc , where $co_assoc(i, j)$ indicates the fraction of times the pattern pair (i, j) is assigned to the same cluster among N clusterings. The final data partition is obtained by applying the single-link method over this similarity matrix, using a fixed threshold, t .

Figure 12 presents results of the combination of data partitions, accord-

ing to the evidence accumulation framework, on the contour images data set (dendrograms produced by the single-link method over the *co_assoc* matrix are shown). In figure 12(a) we combined results produced by the SL-NED, ECP-NED, RDGC and MCL clustering algorithms using the voting mechanism proposed in [17]; by thresholding on this dendrogram we can identify the clusters formed by: (1)- tool 1; (2)- tool 2, pose closed; (3)- tool 2, pose half-opened, and tool 3. In a second experiment, more clusterings were added: the NED, ECP, RDGC and NRDCL similarity matrices between string patterns were used in a complete-link clustering strategy. The dendrogram resulting from this combination is shown in figure 12(b). We can now identify a more clear separation between classes. By adequate thresholding on the dendrogram, one obtains four clusters: (1)- tool 1; (2)- tool 2, pose closed; (3)- tool 2 and tool 3, pose half-opened; (4)- tool 3, pose wide-opened.

The combination of clusterings is therefore a new direction in cluster analysis, that is worth further investigation.

Acknowledgments

This work was partially supported by the Portuguese Foundation for Science and Technology (FCT), Portuguese Ministry of Science and Technology, and FEDER, under grant POSI/33143/SRI/2000.

References

- [1] T. A. Bailey and R. Dubes. Cluster validity profiles. *Pattern Recognition*, 15(2):61–83, 1982.
- [2] J. Buhmann and M. Held. Unsupervised learning without overfitting: Empirical risk approximation as an induction principle for reliable clustering. In Sameer Singh, editor, *International Conference on Advances in Pattern Recognition*, pages 167–176. Springer Verlag, 1999.
- [3] H. Bunke. String matching for structural pattern recognition. In H. Bunke and A. Sanfeliu, editors, *Syntactic and Structural Pattern Recognition, Theory and Applications*, pages 119–144. World Scientific, 1990.

- [4] H. Bunke. Recent advances in string matching. In H. Bunke, editor, *Advances in Structural and Syntactic Pattern Recognition*, pages 107–116. World Scientific, 1992.
- [5] M. Chavent. A monothetic clustering method. *Pattern Recognition Letters*, 19:989–996, 1998.
- [6] J-Y. Chen, C. A. Bouman, and J. P. Allebach. Fast image database search using tree-structured VQ. In *Proc. of IEEE Int’l Conf. on Image Processing*, volume 2, pages 827–830, 1997.
- [7] J. M. Coggins. Dissimilarity measures for clustering strings. In D. Sankoff and J. Kruskal, editors, *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, chapter 1, pages 311–321. Reprint, with a forward by J. Nerbonne, Stanford, CA: CLSI Publications, [1983] 1999.
- [8] G. Cortelazzo, D. Deretta, G. A. Mian, and P. Zamperoni. Normalized weighted levenshtein distance and triangle inequality in the context of similarity discrimination of bilevel images. *Pattern Recognition Letters*, 17:431–436, 1996.
- [9] G. Cortelazzo, G. A. Mian, G. Vezzi, and P. Zamperoni. Trademark shapes description by string-matching techniques. *Pattern Recognition*, 27(8):1005–1018, 1994.
- [10] R. Dubes and A. K. Jain. Validity studies in clustering methodologies. *Pattern Recognition*, 11:235–254, 1979.
- [11] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, second edition, 2001.
- [12] Y. El-Sonbaty and M. A. Ismail. On-line hierarchical clustering. *Pattern Recognition Letters*, pages 1285–1291, 1998.
- [13] J. Barros et al. Using the triangle inequality to reduce the number of computations required for similarity-based retrieval. In *Proc. of SPIE/IS&T, Conference on Storage and Retrieval for Still Image and Video Databases IV*, volume 2670, pages 392–403, 1996.
- [14] M. Figueiredo and A. K. Jain. Unsupervised learning of finite mixture models. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24(3):381–396, 2002.

- [15] W. B. Frakes. Stemming algorithms. In William B. Frakes and Ricardo Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*, chapter 8, pages 131–160. Prentice Hall, 1992.
- [16] A. L. Fred. Clustering of sequences using a minimum grammar complexity criterion. In *Grammatical Inference: Learning Syntax from Sentence*, pages 107–116. Springer-Verlag, 1996.
- [17] A. L. Fred. Finding consistent clusters in data partitions. In Josef Kittler and Fabio Roli, editors, *Multiple Classifier Systems*, volume LNCS 2096, pages 309–318. Springer, 2001.
- [18] A. L. Fred and J. Leitão. A minimum code length technique for clustering of syntactic patterns. In *Proc. Of the 13th IAPR Int'l Conference on Pattern Recognition*, pages 680–684, Vienna, 1996.
- [19] A. L. Fred and J. Leitão. Solomonoff coding as a means of introducing prior information in syntactic pattern recognition. In *Proc. of the 12th IAPR Int'l Conference on Pattern Recognition*, pages 14–18, 1994.
- [20] A. L. Fred and J. Leitão. A comparative study of string dissimilarity measures in structural clustering. In Sameer Singh, editor, *Int'l Conference on Advances in Pattern Recognition*, pages 385–384. Springer, 1998.
- [21] A. L. Fred and J. Leitão. Clustering under a hypothesis of smooth dissimilarity increments. In *Proc. of the 15th Int'l Conference on Pattern Recognition*, volume 2, pages 190–194, Barcelona, 2000.
- [22] A. L. Fred, J. S. Marques, and P. M. Jorge. Hidden markov models vs syntactic modeling in object recognition. In *Int'l Conference on Image Processing, ICIP'97*, pages 893–896, Santa Barbara, October 1997.
- [23] K. S. Fu. Syntactic pattern recognition. In *Handbook of Pattern Recognition and Image Processing*, pages 85–117. Academic Press, 1986.
- [24] K. S. Fu and S. Y. Lu. A clustering procedure for syntactic patterns. *IEEE Trans. Systems Man Cybernetics*, 7(7):537–541, 1977.
- [25] K. S. Fu and S. Y. Lu. Grammatical inference: Introduction and survey -part I and II. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(5):343–359, 1986.

- [26] J. A. Garcia, J. Valdivia, F. J. Cortijo, and R. Molina. A dynamic approach for clustering data. *Signal Processing*, 2:181–196, 1995.
- [27] M. Har-Even and V. L. Brailovsky. Probabilistic validation approach for clustering. *Pattern Recognition*, 16:1189–1196, 1995.
- [28] D. Harman. Ranking algorithms. In William B. Frakes and Ricardo Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*, chapter 14, pages 363–392. Prentice Hall, 1992.
- [29] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, London, 1979.
- [30] Q. Huang, Z. Liv, and A. Rosenberg. Automated semantic structure reconstruction and representation generation for broadcast news. In *Proc. of IS&T/SPIE Conference on Storage and Retrieval for Image and Video Databases VII*, pages 50–62, 1999.
- [31] A. Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall, 1989.
- [32] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.
- [33] A.K. Jain, M. N. Murty, and P.J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, September 1999.
- [34] A. Juan and E. Vidal. Fast median search in metric spaces. In A. Amin and D. Dori, editors, *Advances in Pattern Recognition*, pages 905–912. Springer-Verlag, 1998.
- [35] J. A. Kaandorp. *Fractal Modelling: Growth and Form in Biology*. Springer-Verlag, 1994.
- [36] R. L. Kashyap and B. J. Oommen. String correction using probabilistic models. *Pattern Recognition Letters*, pages 147–154, 1984.
- [37] J. Kittler. Pattern classification: Fusion of information. In Sameer Singh, editor, *Int. Conf. on Advances in Pattern Recognition*, pages 13–22. Springer, 1998.
- [38] J. Kittler, M. Hatef, R. P. Duin, and J. Matas. On combining classifiers. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

- [39] T. Kohonen. Median strings. *Pattern Recognition Letters*, 3:309–313, 1985.
- [40] R. Kothari and D. Pitts. On finding the number of clusters. *Pattern Recognition Letters*, 20:405–416, 1999.
- [41] J. Kruskal. An overview of sequence comparison. In D. Sankoff and J. Kruskal, editors, *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, chapter 1, pages 1–44. Reprint, with a forward by J. Nerbonne, Stanford, CA: CLSI Publications, [1983] 1999.
- [42] S. Y. Lu and K. S. Fu. Stochastic error-correcting syntax analysis for the recognition of noisy patterns. *IEEE Trans. Computers*, C-26(12):1268–1276, December 1977.
- [43] S. Y. Lu and K. S. Fu. A sentence-to-sentence clustering procedure for pattern analysis. *IEEE Trans. Systems Man Cybernetics*, 8(5):381–389, 1978.
- [44] Y. Man and I. Gath. Detection and separation of ring-shaped clusters using fuzzy clusters. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 16(8):855–861, 1994.
- [45] C. D. Martinez-Hinarejos, A. Juan, and F. Casacuberta. Use of median string for classification. In *Proc. of the 15th Int'l Conf. on Pattern Recognition*, pages 907–910, 2000.
- [46] A. Marzal and E. Vidal. Computation of normalized edit distance and applications. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2(15):926–932, 1993.
- [47] G. McLachlan and K. Basford. *Mixture Models: Inference and Application to Clustering*. Marcel Dekker, New York, 1988.
- [48] L. Miclet. Grammatical inference. In H. Bunke and A. Sanfeliu, editors, *Syntactic and Structural Pattern Recognition - Theory and Applications*, pages 237–290. Scientific Publishing, 1990.
- [49] B. Mirkin. Concept learning and feature selection based on square-error clustering. *Machine Learning*, 35:25–39, 1999.
- [50] T. Oates, L. Firoiu, and P. R. Cohen. Using time warping to bootstrap HMM-based clustering of time series. In R. Sun and C. L. Giles, editors,

Sequence Learning: Paradigms, Algorithms and Applications, volume LNAI of *Lecture Notes in Computer Science*, pages 35–52. Springer-Verlag, 2000.

- [51] B. J. Oomen and R. S. K. Loke. Pattern recognition of strings containing traditional and generalized transposition errors. In *Int'l Conf. on Systems, Man and Cybernetics*, pages 1154–1159, 1995.
- [52] B. J. Oommen. Recognition of noisy subsequences using constrained edit distances. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 9(5):676–685, 1987.
- [53] N. R. Pal and J. C. Bezdek. On cluster validity for the fuzzy c-means model. *IEEE Trans. Fuzzy Systems*, 3:370–379, 1995.
- [54] E. J. Pauwels and G. Frederix. Finding regions of interest for content-extraction. In *Proc. of IS&T/SPIE Conference on Storage and Retrieval for Image and Video Databases VII*, volume SPIE Vol. 3656, pages 501–510, San Jose, January 1999.
- [55] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. of IEEE*, 77(2):257–285, 1989.
- [56] E. S. Ristad and P. N. Yianilos. Learning string-edit distance. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(5):522–531, 1998.
- [57] S. Roberts, D. Husmeier, I. Rezek, and W. Penny. Bayesian approaches to gaussian mixture modelling. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20(11), 1998.
- [58] D. Sankoff and J. Kruskal. *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Reprint, with a forward by J. Nerbonne, Stanford, CA: CLSI Publications, [1983] 1999.
- [59] S. Santini and R. Jain. Similarity is a geometer. *Multimedia Tools and Applications*, 5(3):277–306, 1997.
- [60] P. Sebastiani, M. Ramoni, and P. Cohen. Sequence clustering via bayesian clustering by dynamics. In R. Sun and C. L. Giles, editors, *Sequence Learning: Paradigms, Algorithms and Applications*, number 1828 in *Lecture Notes in Computer Science*, pages 11–34. Springer-Verlag, 2000.

- [61] P. Smyth. Clustering sequences with hidden Markov models. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing 9*, pages 72–93. MIT Press, Cambridge, 1997.
- [62] R. J. Solomonoff. A formal theory of inductive inference (part I and II). *Information and Control*, 7:1–22,224–254, 1964.
- [63] D. Stanford and A. E. Raftery. Principal curve clustering with noise. Technical report, University of Washington, <http://www.stat.washington.edu/raftery>, 1997.
- [64] E. Tanaka. Parsing and error correcting parsing for string grammars. In *Syntactic and Structural Pattern Recognition – Theory and Applications*, pages 55–84. Scientific Publishing, 1990.
- [65] H. Tenmoto, M. Kudo, and M. Shimbo. MDL-based selection of the number of components in mixture models for pattern recognition. In Adnan Amin, Dov Dori, Pavel Pudil, and Herbert Freeman, editors, *Advances in Pattern Recognition*, volume 1451 of *Lecture Notes in Computer Science*, pages 831–836. Springer Verlag, 1998.
- [66] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, 1999.
- [67] C. Zahn. Graph-theoretical methods for detecting and describing gestalt structures. *IEEE Trans. Computers*, C-20(1):68–86, 1971.