A COMPARATIVE STUDY OF STRING DISSIMILARITY MEASURES IN STRUCTURAL CLUSTERING

Ana L. N. Fred, Instituto Superior Técnico, Lisbon, Portugal
José M. N. Leitão, Instituto Superior Técnico, Lisbon, Portugal

ABSTRACT

This paper addresses structural clustering by stressing the distinction between string matching and structural resemblance. 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; string matching and error-correcting parsing. Test examples include: synthetic data, with variable length strings; text analysis; and contour images.

INTRODUCTION

Clustering is a powerful tool in revealing the intrinsic organization of data. Concerning structural patterns, it consists of an unsupervised association of data based on the similarity of their structures and primitives. Potential applications of structural clustering are: unsupervised grammatical inference; text analysis; contour image categorisation.

Clustering algorithms for structural pattern analysis based on string descriptions are extensions of conventional clustering methods [1] by introducing dissimilarity measures between strings. Distances and dissimilarity measures commonly found in the literature are based on string matching [2, 3], assuming string edit operations: insertion, substitution and deletion of symbols, to which costs are associated; typically, they are variations of the Levensthein distance, of which the probabilistic modeling is a particular instance. These measures are applied, for instance, in error correction of noisy sentences [4, 5] and in recognition tasks [6, 7, 8, 9, 10]. Recently we have proposed similarity measures based on the search of common subpatterns [11] or composition rules [12], exploring the concept of minimum description. The basic idea is that, if strings are structurally similar, then the description of the ensemble should be more compact than the description of the strings when considered individually. Reference [11] concerns the search of common subpatterns based on Solomonoff's coding [13, 14], the similarity between strings being defined as a ratio of decrease in code length. In [12] strings structure is modelled by grammars; similar rules of composition lead to a reduction in the global grammar complexity, which is the basis of the proposed similarity measure between strings. The later measures are then extended to sets of strings, constituting the cluster formation rules used in hierarchical agglomerative clustering algorithms. Other clustering strategies found in the literature include: *sentence to sentence clustering*, described in [16, 17], based on the comparison of a candidate string with sentences in previously formed clusters (clustering based on a nearest-neighbor rule) or with cluster centre strings (cluster centre technique); grammatical inference and error-correcting parsing are combined in a procedure described in [15, 17], where grammars characterize the structural identity of the formed clusters and the distance between an input sequence and a language is computed by error-correcting parsing.

This paper addresses structural clustering by stressing the distinction between string matching and structural resemblance. The similarity measures proposed by the authors are analyzed

and compared with string matching and error-correcting parsing techniques in the context of structural clustering. Hierarchical agglomerative clustering algorithms using these proximity measures and Fu's algorithm [15] are used in the analysis.

The paper is organized as follows. In a first section the set of dissimilarity measures is defined and analyzed in light of a short example. The second section presents the clustering algorithms; in the next section their performance is evaluated in several test examples in terms of the correctness of the associations made. A final section summarizes the conclusions.

STRINGS, DISTANCES AND DISSIMILARITY MEASURES

A distance between strings s_i and s_j is some function $d(s_i, s_j)$ measuring the dissimilarity between the two strings that obey the following properties:

1. Symmetry: $d(s_i, s_j) = d(s_j, s_i)$.
2. Triangle inequality: $d(s_i, s_l) \leq d(s_i, s_j) + d(s_j, s_l)$.

This paper will focus on four approaches to measure string dissimilarity: (i)- string matching based on error transformations; (ii) - structure modeling by grammars, resemblance being measured by grammatical complexity; (iii) - searching for common subpatterns in a minimum code length framework; (iv) - distances between a string and a language described by a grammar, by means of error-correcting parsing.

DISSIMILARITY BASED ON STRING EDIT OPERATIONS

This approach is based on the definition of string editing operations or error transformations, namely substitution, deletion and insertion of symbols. The well known *Levensthein distance* between two strings s_1 and s_2 , $d_L(s_1, s_2)$, is defined as the minimum number of editing operations needed to transform s_1 into s_2. An extension of the Levensthein distance by associating different costs to the several editing operations is known as the *weighted Levensthein distance*, defined as

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

where $\gamma(T) = \sum_{i=1} \gamma(T_i)$, T_i is the ith editing operation performed in the transformation T, and $\gamma(T_i)$ is the cost associated with that operation.

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 (see for instance [8] and the references therein); thus it is a metric.

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

Normalization of the previous distances with respect to string lengths are obtained by post-normalization ($d_{NW}(s_1, s_2) = d_W(s_1, s_2)/max\{|s_1|, |s_2|\}$) - normalized weighted Levensthein distance, and by optimal minimization of the distance normalized by the length of the editing path - normalized edit distance [6]:

$$d_{NE}(s_1, s_2) = min\left\{\frac{\gamma(T)}{|T|} \;\middle|\; T \text{ is an editing path between } s_1 \text{ and } s_2\right\}$$

where $|T|$ is the length of the editing path. Vidal [6] 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].

DISSIMILARITY BASED ON ERROR-CORRECTING PARSING

Fu [15, 17] defined a distance between strings based on the modeling of string structure by means of grammars and on the concept of error-correcting parsing (ECP). 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):

$$d_{ECP_f}(s_1, s_2) = \overset{min}{y} \{d_W(s_1, y) | y \in L(G_2) \text{ and } G_2 := \text{ Grammar inferred from } s_2\}$$

The computation of the dissimilarity between strings requires two steps: 1- modeling of strings with grammars; 2- computation of the dissimilarity by error-correcting parsing. 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 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 hint about the underlying structure.

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, using Crespi-Reghizzi's method [19, 20] for grammatical inference one obtains: $d_{ECP_f}(dfg, (abc)^2) = 3$ and $d_{ECP_f}((abc)^2, dfg) = 6$. In order to preserve the symmetry property we will therefore use the new definition:

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

In this paper we will consider another variation of the previous measure by using the normalized edit distance, which we designate by d_{NECP}. Neither d_{ECP} or d_{NECP} constitute a metric because they do not obey the triangle inequality (for instance, $d_{ECP}((dfg)^8, (abc)^2) = 24, d_{ECP}((abc)^2, dfg) = 3$ and $d_{ECP}(dfg, (dfg)^8) = 0$).

GRAMMAR COMPLEXITY-BASED DISSIMILARITY

In [12] we have proposed a new measure of similarity between strings according to which patterns are described by grammars, and exploring the concept of grammar complexity. The basic idea is that, if two sentences are structurally similar, than their joint description will be more compact than their isolated description due to sharing on common rules of symbol composition; the compactness of representation is quantified by the grammar complexity and the similarity is measured by the *ratio of decrease in grammar complexity*, as follows:

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

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

The complexity of a grammar G, $C(G)$, is defined as [12]

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

where α_{ij} represents the right side of the jth production for the ith non-terminal symbol of the grammar, and

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

with k_i being the number of times that the symbol a_i appears in α, and n is the length of the grammatical sentence α.

Results obtained for the similarity measure, as for the preceding case, depend on the grammatical inference strategies adopted. However, the following characteristics are independent of the method adopted:

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

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

MINIMUM CODE LENGTH-BASED SIMILARITY

In [11] we proposed another measure of structural similarity exploring the notion of compressibility of sequences and algorithmic complexity. Solomonoff's code is there used for the search of pattern regularities and sequence compression. According to this coding scheme a sequence s_i is represented by the triplet: $alphabet,\ symbol\_definition,\ s_i\_coded$, 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 single symbols, and rewriting the sequences using them [11, 13]. Compact codes are produced when sequences exhibit local or distant inter symbol interactions. 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 we designate by NRDCL:

$$NRDCL(s_1, s_2) = \frac{code\_len(s_1) + code\_len(s_2) - code\_len(s_1, s_2)}{(as)/2 + abs(|s_1\_coded| - |s_2\_coded|)}$$

with

$$as = |alphabet\_s_1| + |symbol\_definition\_s_1| + |alphabet\_s_2| + |symbol\_definition\_s_2|$$

This similarity measure is symmetric but the triangle inequality does not hold in general.

EXAMPLE

Table 1 illustrates the previous proximity measures. As shown, d_L and d_{NE} are typically conditioned by the similarity of strings lengths, higher similarity being found between $(abc)^8$ and $(cba)^8$ than with $(abc)^{48}$. They serve the purpose of string matching rather than structural similarity. This effect is compensated in the remaining measures with true independence in the strings lengths being obtained with the ECP and RDGC measures - grammatical models. The RDGC similarity gives higher penalization to the situation of distinct alphabets; the ECP method enables the establishment of differentiated costs, but criteria must be defined for the construction of these weights.

ID	String	d_L	d_{NE}	d_{ECP}	d_{NECP}	$RDGC$	$NRDCL$
1	$(dfg)^8$	24	1	24	1	0	0
2	$(dfg)^{18}$	54	1	24	1	0	0
3	$(abc)^8$	0	0	0	0	1	1
4	$(abc)^{18}$	30	0.556	0	0	1	0.878
5	$(abc)^{48}$	120	0.833	0	0	1	0.818
6	$(cba)^8$	16	0.667	13	0.542	0	0
7	$(cba)^{18}$	30	0.556	13	0.519	0	0.146
8	$(abc)^8(cba)^8$	24	0.500	1	0.042	0.586	0.591
9	$(abc)^2(cba)^{14}$	24	0.500	1	0.042	0.586	0.210
10	$(abcf)^7$	16	0.400	8	0.250	0	0.429

Table 1: Dissimilarity and similarity values between string $(abc)^8$ and the strings on the left column.

CLUSTERING

The clustering algorithms proposed by the authors in [11, 12] integrate the RDGC and the NRDCL similarity measures into the hierarchical agglomerative clustering framework. This clustering paradigm can be summarized as follows: begin by assigning each sample to a single cluster and continue by merging clusters based on the similarity between clusters until a one-cluster solution is produced. The algorithm uses a proximity matrix measuring the similarity / dissimilarity between all pairs of clusters at a given step. It produces a sequence of partitions of the original data. A particular clustering is obtained by either fixating the number of clusters desired or by comparing the proximity values between clusters with a threshold; in this case the number of classes obtained is not known *a priori*. A schematic description of the algorithm can be expressed as:

- *Algorithm:* Hierarchical Agglomerative Clustering.
- *Input:* A set of strings $X = \{s_1, s_2, \ldots, s_n\}$; a threshold th or the desired number of clusters N_c.
- *Output:* A partition of X into m clusters ($m = N_c$ if the latter is specified), $C_1, C_2, \ldots, C_m$.
- *Steps:*
  1. Set each pattern into a single cluster. Set $N_a = n$.
  2. compute the proximity matrix between all pairs of strings.
  3. If $N_a = N_c$ go to step 6.

4. Select from the proximity matrix the pair of clusters C_i, C_j most similar and let $simil$ be that value of similarity. If $simil$ is less than th (greater than th if it is a dissimilarity measure) than go to step 6.
5. Merge the two clusters and update the proximity matrix accordingly. Set $N_a = N_a - 1$ and continue in step 3.
6. End the computation by returning the data partition encountered.

Let C_i be a cluster represented by a set of n_i sentences $X_i = \{s_1^i, s_2^i \ldots, s_{ni}^i\}$. Distinct clustering algorithms are obtained by appropriate definition of the dissimilarity between strings and the rule of cluster formation - proximity function between clusters.

For the definition of the proximity matrix, the dissimilarity between pairs of strings are extrapolated for sets of strings. Table 2 (a) summarizes the characteristics of the algorithms analyzed. In the first one, distances are based on string editing operations and the nearest-neighbor (NN) rule is applied in cluster formation; it can be viewed as a hierarchical version of the sentence-to-sentence clustering procedure proposed in [16] by considering both the Levensthein and normalized edit distances. It may be shown that it has the same time complexity, being less sensitive to the order of presentation of the patterns.

a) Hierarchical Agglomerative Schemes			
Philosophy of the method	Proximity between clusters	String similarity	Acronym
String edit operations	Nearest-neighbor rule: $d(C_i, C_j) = min\{d(s_k, s_l) \mid s_k \in C_i, s_l \in C_j\}$	d_L d_{NE}	NN-SEO-DL NN-SEO-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: $NRDCL(C_i, C_j) = = \frac{code\_len(C_i) + code\_len(C_j) - code\_len(C_i, C_j)}{(as)/2 + abs(\lvert C_i\_coded\rvert - \lvert C_j\_coded\rvert)}$ $as = \sum_{k=i,j} \lvert alph\_C_k\rvert + \lvert symb\_def\_C_k\rvert$	$NRDCL$	MCL
b) Sentence to Sentence Clustering			
Error correcting parsing	Nearest-neighbor rule: $d(s_i, C_j) = min\{d(s_i, s_l) \mid s_l \in C_j\}$	d_{ECP} d_{NECP}	ECP-DL ECP-NED

Table 2: Clustering algorithms.

Finally, as shown in table 2(b), the study will also include the partitional clustering algorithm based on error-correcting parsing described in [15, 17]. This algorithm assumes grammars to model cluster identity, starting with a single cluster with a first sample; the remaining data is classified by computing the distance defined in the previous section between the candidate string and 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.

TEST EXAMPLES

The above measures are evaluated and compared in the clustering of test cases and real data. Concerning grammatical inference, Crespi-Reghizzi's [19, 20] method is used. Evaluation of

the methods is based on the *classification error*, which measures the ability of the classifier to correctly associate patterns in clusters.

TEST STRINGS

Figure 1 presents the results of clustering of the test strings defined in section . As shown, grammar based methods (MGC and ECP) perfectly associate patterns at very high similarity (low dissimilarity) values. The MCL method also achieves the perceptual intuitive clusters. Concerning the string matching algorithms, neither the normalized or the unnormalized versions were able to perform reasonable associations, which are mostly dictated by string length proximity.

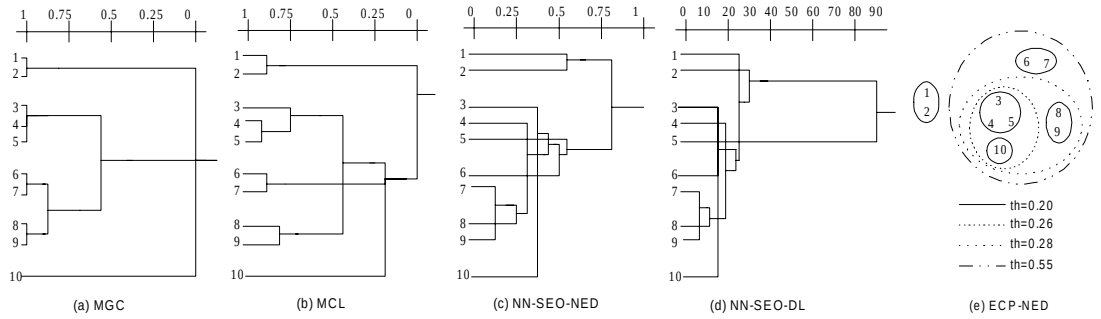

Figure 1: (a) to (d) - Dendrograms for the hierarchical methods. (e) - Clusters obtained with the ECP-NED method for increasing values of the parameter th. Results obtained with ECP-DL are similar, with the particularity that, by increasing the threshold th, a jump from four to two clusters is observed.

CLUSTERING OF CONTOUR IMAGES

In this section we illustrate the capacity of the several clustering algorithms described previously in the partitioning of images of objects of hardware tools based on a differential 8-directional chain code (a more detailed description of the contour definition process and the images database can be found in [18]). Three tools are analyzed, 10 samples per tool. Table 3 illustrates the typical patterns.

Class	Tool	Samples
1		6107262002627016370262002620726107260162702620026
		7117262002620016270163702620726107361072611637026
		7117262002627026200262073701637026200262002620016
2		0000000066000000171000006000001000000007500000000
		0000000050000000000000760000016100000066000000003
		0000000050000000171000005000001610000007600000003
3		1000000005000000007207000701700000000050000000012
		1000000005000000000107000702700000000050000000012
		1000000005700000000710700770270000000005000000012

Table 3: Sample string contour descriptions of hardware tools.

As the length of the strings is constant, the normalized and not normalized versions of the algorithms based on string edit operations provide the same cluster associations. As shown in figure 2(a) the MGC method completely separates the three classes of patterns. The NN rules are good in associating the patterns of classes 2 and 3 and in distinguishing between pattern

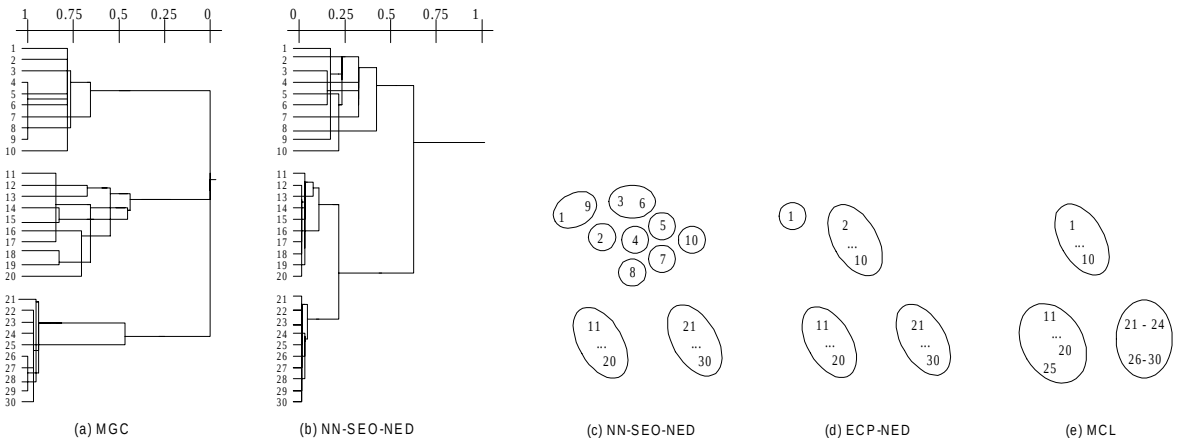

Figure 2: Dendrograms for the MGC method (a) and NN-SEO-NED (b). Clusters obtained with NN-SEO-NED ($th = 0.2$) (c), ECP-NED ($th = 0.1$) (d) and MCL ($th = 0.48$) (e) methods.

1 and the remaining; however, separation between classes 2 and 3 is not possible without fragmentation of class 1 into several (mostly single) clusters, as illustrated in figure 2(c). Error-correcting parsing refines this result by grouping all but one pattern from class 1 into one cluster; the inclusion of sample number 1 into cluster 1 implies the union of samples 11 to 30. The MCL method failed to associate pattern 25 in the correct class.

ORGANIZING EMAIL

The next example consists of the first lines of text extracted from email messages, as shown in table 4, by removing space characters. Variable length samples using the English and the Portuguese language were included.

ID	Mail Type	Cl.	Lang.	# chars	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 RENSSELAER'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 advertisement	3	Port.	153	PROVAS DE MESTRADO EM ENGENHARIA . . .
9	MSc advertisement	3	Port.	280	PROVAS DE MESTRADO EM ENGENHARIA . . .
10	MSc advertisement	3	Port.	168	PROVAS DE MESTRADO EM ENGENHARIA . . .

Table 4: Text from email messages.

In this case, only the MCL method was able to provide meaningful associations, 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 (below 10e-2)were found by the MGC method at the character level, no significant associations being made. Methods based on error transformations failed to discover common patterns in the data. The normalized dissimilarity values of the NN method were high and in narrow ranges, resulting in no associations for threshold values above 0.73; when a fixed number of

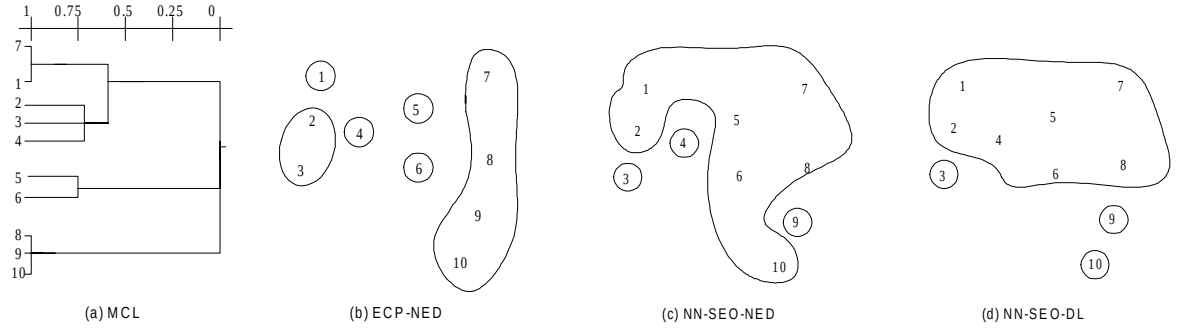

Figure 3: (a) Dendrogram for the MCL method. (b) Associations obtained with the ECP-NED method for $th = 0.5$. (c) - (d) Examples of associations provided by the NN rule when setting the number of clusters to 4. The dissimilarities ranges are, respectively, 0.73-0.76 and 90-214.

classes was imposed, regardless of the dissimilarity measures values, arbitrary associations were provided by the several methods (see a few examples in figure 3). Although several of the sentences have common sequences, resemblance cannot be measured in terms of string matching as the order of the narrative is arbitrary.

CONCLUSIONS

In this paper structural clustering of patterns described in the string format was addressed in a comparative study of four classes of dissimilarity measures. These included distances based on string editing operations, namely the Levensthein distance, d_L, between two strings and its normalization by the length of the editing path, d_{NE}. The dissimilarity measures d_{ECP} and d_{NECP} considered the minimum (normalized) Levensthein distance between one string and the language generated by the grammar inferred from the second string, computed by error-correcting parsing. Finally, the similarity measures $RDGC$ and $NRDCL$ addressed the concepts of grammatical complexity and minimum description length, exploring, respectively, the structural resemblance between strings expressed in terms of common rules of description and common patterns of statistical dependencies between symbols manifesting as similar subsequences.

The above measures were evaluated in the categorisation of synthetic and real data, integrated in hierarchical agglomerative and partitional clustering algorithms. Test examples included the analysis of email messages and clustering of contour images.

The major conclusions are summarized as follows:

- Minimum description-based methods are best suited for context analysis, where the appearance of certain sequences rather than the particular order of their appearance is of major importance.
- String matching techniques are best suited for exact recognition of patterns.
- Grammatical inference-based methods are more reliable in grouping of variable length patterns with regular structures. Additionally, they have the ability to reduce large collections of data into lower dimension models that preserve the underlying structure of the original collection, thus providing a means for data reduction.

ACKNOWLEDGEMENTS

This work was partially supported by the projects PECS/C/SAU/212/95 and PRAXIS 2/2.1/ TIT/1580/95.

REFERENCES

[1] Jain AK, Dubes RC. Algorithms for Clustering Data. Prentice Hall, Englewood Cliffs, 1988.

[2] Bunke H. String matching for structural pattern recognition. In: Bunke H, Sanfeliu A (eds). Syntactic and Structural Pattern Recognition, Theory and Applications. World Scientific, 1990, pp 119–144.

[3] Bunke H. Recent advance in string matching. In: Bunke H (eds). Advances in Structural and Syntactic Pattern Recognition. World Scientific, 1992, pp 107–116.

[4] Kashiap RL, Oommen BJ. String correction using probabilistic models. Pattern Recog. Letters 1984; 147–154.

[5] Oommen BJ. Recognition of noisy subsequences using constrained edit distances. IEEE Trans. Pattern Anal. and Machine Intelligence 1987; PAMI-9(5): 676–685.

[6] Marzal A, Vidal E. Computation of normalized edit distance and applications. IEEE Trans. Pattern Anal. and Machine Intelligence 1993; PAMI-2(15): 926–932.

[7] Bunke H, Buhler U. 2-D Invariant shape recognition using string matching. In Proc. 2nd Int. Conf. on Automation, Robotics and Computer Vision, 1992.

[8] Cortelazzo G, Deretta D, Mian GA, Zamperoni P. Normalized weighted Levensthein distance and triangle inequality in the context of similarity discrimination of bilevel images. Pattern Recognition Letters 1996; 17: 431–436.

[9] Cortelazzo G, Mian GA, Vezzi G, Zamperoni P. Trademark shapes description by string-matching techniques. Pattern Recognition 1994; 27(8): 1005–1018.

[10] Peng HL, Chen SY. Trademark shape recognition using closed contours. Pattern Recognition Letters 1997; 18: 791–803.

[11] Fred ALN, Leitão JMN. A minimum code length technique for clustering of syntactic patterns. In Proc. of the 13th IAPR Int'l Conference on Pattern Recognition, vol 2. IEEE Press, 1996, pp 680–684.

[12] Fred ALN. Clustering of sequences using a minimum grammar complexity criterion. In: Miclet L, Higuera C (eds). Grammatical Inference: Learning Syntax from Sentences. Springer-Verlag, 1996, pp 107–116.

[13] Solomonoff RJ. A formal theory of inductive inference (part I and II). Information and Control 1964; 7:1-22, 224–254.

[14] Fred ALN, Leitão JMN. 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, vol 2. IEEE Press, 1994, pp 14–18.

[15] Fu KS, Lu SY. A clustering procedure for syntactic patterns. IEEE Trans. Systems Man Cybernetics 1977; SMC-7(7):537–541.

[16] Lu SY, Fu KS. A sentence-to-sentence clustering procedure for pattern analysis. IEEE Trans. Systems Man Cybernetics 1978; SMC-8(5):381–389.

[17] Fu KS. Syntactic pattern recognition. In Handbook of Pattern Recognition and Image Processing. Academic Press, 1986, pp 85–117.

[18] Fred ALN, Marques JS, Jorge PM. Hidden Markov models vs syntactic modeling in object recognition. In Proc. 1997 International Conference on Image Processing, IEEE Press, 1997, pp 893–896.

[19] Fu KS, Lu SY. Grammatical inference: introduction and survey -part I and II. IEEE Trans. Pattern Anal. and Machine Intelligence 1986; PAMI-8(5): 343–359.

[20] L. Miclet. Grammatical inference. In: Bunke H, Sanfeliu A (eds). Syntactic and Structural Pattern Recognition - Theory and Applications. Scientific Publishing, 1990, pp 237–290.