Mining Unstructured Data via Computational Intelligence

Angel Kuri-Morales

Instituto Tecnológico Autónomo de México
Río Hondo No. 1
México 01000, D.F.
México
akuri@itam.mx

Abstract. At present very large volumes of information are being regularly produced in the world. Most of this information is unstructured, lacking the properties usually expected from, for instance, relational databases. One of the more interesting issues in computer science is how, if possible, may we achieve data mining on such unstructured data. Intuitively, its analysis has been attempted by devising schemes to identify patterns and trends through means such as statistical pattern learning. The basic problem of this approach is that the user has to decide, a priori, the model of the patterns and, furthermore, the way in which they are to be found in the data. This is true regardless of the kind of data, be it textual, musical, financial or otherwise. In this paper we explore an alternative paradigm in which raw data is categorized by analyzing a large corpus from which a set of categories and the different instances in each category are determined, resulting in a structured database. Then each of the instances is mapped into a numerical value which preserves the underlying patterns. This is done using a genetic algorithm and a set of multi-layer perceptron networks. Every categorical instance is then replaced by the adequate numerical code. The resulting numerical database may be tackled with the usual clustering algorithms. We hypothesize that any unstructured data set may be approached in this fashion. In this work we exemplify with a textual database and apply our method to characterize texts by different authors and present experimental evidence that the resulting databases yield clustering results which permit authorship identification from raw textual data.

Keywords. Data bases, Neural Networks, Genetic Algorithms, Categorical encoding.

1. Introduction.

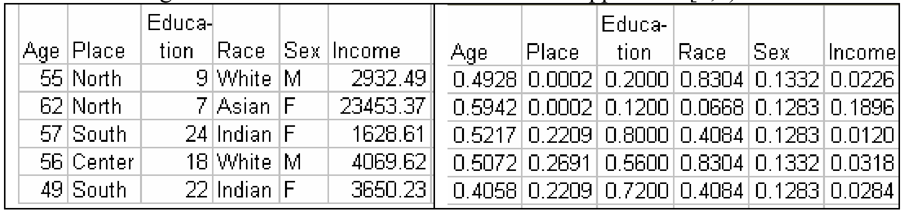
The problem of analyzing large sets of unstructured data sets has grown in importance over the last years. As of April 2014 there was an estimated 958,919,789 sites in the world [1]. Due to corrections to Metcalfe's law [2], which state that the value of a telecommunications network is proportional to $n\ (\log_2 n)$ of the number of connected users of the system, there is a network world value of O(28.9e+09). The associated amount of data generated may be inferred from this number and even conservative metrics yield very large estimates. Most of these data are unstructured and recent commercial [3] approaches to the problem attest to the increasing importance assigned to this fact. In the computer science community, data mining of texts [4], music [5] and general information [6], [7] is being approached with growing interest. In the vast majority of the cases, information extraction (IE) is highlighted and emphasizes the use of anaphora, the use of an expression the interpretation of which depends upon another expression in context. This approach, although intuitive and natural, suffers from the obvious disadvantage of being case-based. That is, a method developed for, say, texts in English will not be directly applicable to other languages and much less to other kinds of information. For example, even when limiting our range of study to texts stemming from corporate finance (i.e. mining industry literature for business intelligence), "horizontal" test mining (i.e. patent analysis) or life sciences research (i.e. mining biological pathway information) every one of the lines just mentioned relies on a case-by-case determination of the anaphoric usage.

The problem at the very heart of this issue is the fact that we must preserve the patterns underlying the information and it had not been treated with success in the past. In [8], however, a method to encode non-numerical data in mixed (numerical and categorical) data bases was shown to be effective in preserving the embedded patterns. In what follows, we denote with *G* the number of generations of a genetic algorithm (EGA), with *n* the number of attributes in the database. MLP denotes a multilayered perceptron network. MD denotes a data base where some of the attributes *may* be numerical and others *are* categorical. ND denotes the data base where all instances of the categories have been replaced by a numerical code. The algorithm developed therein was dubbed CENG and may be succinctly described as follows.

```
CENG: An algorithm for categorical encoding
  1. Specify the mixed database MD.
  2. MD is analyzed to determine $c$ (the number of categorical attributes) and $t$ (the
       number of instances of all categories). The size of the genome (the number
       of bits needed to encode the solution) is $L=22ct$. EGA (the Eclectic Genetic
       Algorithm [9]) is programmed to minimize the fitness function; it randomly
       generates a set of $PS$ (the number of individuals) strings of size $L$. This is
       population $P_0$. Every one of the $PS$ strings is an individual of EGA.
  for $i=1$ to $G$
    for $j=1$ to $PS$
         From individual $j$, $ct$ numerical codes are extracted and $ND_{ij}$ is generated
       by replacing the categorical instances with their numerical counterparts.
       Numerical variables are left undisturbed.
         $MLP_{ij}$'s architecture (for EGA's individual $j$) is determined.
      for $k=1$ to $n-1$
               $MLP_{ij}$ is fed with a data matrix in which the $k$-th attribute of ND
```

```
                is taken as a variable dependent on the remaining $n-1$. $MLP_{ij}$ is
                trained and the maximum error $e_{ijk}$ (i.e. the one resulting by feeding
                the already trained $MLP_{ij}$ with all tuples vs. the dependent variable)
                is calculated.
          endfor
          Fitness($j$) = max($e_{ijk}$ )
     endfor
      if Fitness($j$)<0.01 the EGA ends. Otherwise the $PS$ individuals of $P_i$ are selected,
      crossed over and mutated yielding the new $P_i$.
   endfor
```

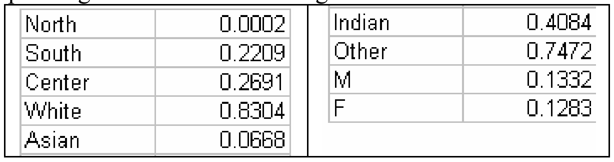
CENG delivers the codes for every one of the *ct* instances of the categorical variables and ND: a version of MD in which all categorical instances are replaced by the proper numerical codes.
An example of a Mixed Database and the resulting Numerical Database after CENG is shown in Figure 1. All numerical values have been mapped into [0,1).



Age	Place	Educa-tion	Race	Sex	Income	Age	Place	Educa-tion	Race	Sex	Income
55	North	9	White	M	2932.49	0.4928	0.0002	0.2000	0.8304	0.1332	0.0226
62	North	7	Asian	F	23453.37	0.5942	0.0002	0.1200	0.0668	0.1283	0.1896
57	South	24	Indian	F	1628.61	0.5217	0.2209	0.8000	0.4084	0.1283	0.0120
56	Center	18	White	M	4069.62	0.5072	0.2691	0.5600	0.8304	0.1332	0.0318
49	South	22	Indian	F	3650.23	0.4058	0.2209	0.7200	0.4084	0.1283	0.0284

Fig. 1. Example of Mixed and Numerical Data after CENG.

The corresponding codes are shown in Figure 2.



North	0.0002	Indian	0.4084
South	0.2209	Other	0.7472
Center	0.2691	M	0.1332
White	0.8304	F	0.1283
Asian	0.0668		

Fig. 2. Code for categorical instances.

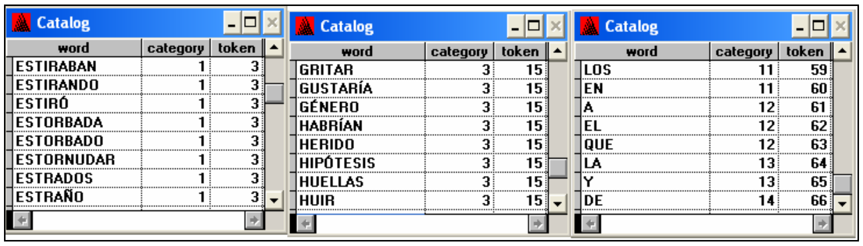
Once having shown that CENG does find the correct codes for categorical attributes in mixed data bases, a corollary is that classic numerical clustering algorithms (such as Fuzzy C-Means [10], Self-Organizing Maps [11] or Entropy Based Clustering [12]) may be used and, furthermore, any text (indeed, any collection of tokenizable data) may be treated as a set of categorical variables provided categories and their corresponding instances are identified.
The rest of the paper is organized as follows. In section 2 we describe the method we applied to tokenize Spanish texts. In section 3 we describe the process of encoding the tokenized data base to obtain the corresponding clusters. In section 4 we present an algorithm developed to identify the correspondence of the clusters in a labeled data base. In section 5 we describe the experiments we performed to test the viability of our method. In section 6 we present our conclusions.

2. Tokenizing Spanish Texts

To tokenize unstructured data we first have to find a representative sample which adequately characterizes the universe of data (*U*). Once having done so we select the number of categories (*c*). The next step is to determine the instances within every category (*t*). The number of tokens (*k*) per tuple determines the form of the structured data base. The final step is to select the text to tokenize and identify the tokens which will populate the structured data base. Notice that these steps are independent of the nature of *U*.

2.1 Tokenizing the Universe of Data

We started by selecting a collection of Spanish texts from different authors: Gabriel García Márquez, Julio Cortázar, Miguel de Cervantes, Jorge Luis Borges and Spanish translations of James Joyce and William Shakespeare. 125,882 words were extracted, from which 15,031 distinct ones were identified (i.e. | *U* | = 15,301). We then made *c*=12 and *t*=4. This was achieved by obtaining the frequencies of every one of the 15,301 words and dividing then into 12 equally distributed intervals. That is, every one of the intervals in category *i* consists of as many words as those needed to account for 1/12 of the total. An index was generated wherein up to 4 equally spaced subintervals were defined. This translates into the fact that there are more words in the initial intervals (more finely split frequencies) whereas those intervals where the more probable words lie consist of less words. An example taken from the actual database is shown in Figure 3.



word	category	token
ESTIRABAN	1	3
ESTIRANDO	1	3
ESTIRÓ	1	3
ESTORBADA	1	3
ESTORBADO	1	3
ESTORNUDAR	1	3
ESTRADOS	1	3
ESTRAÑO	1	3

word	category	token
GRITAR	3	15
GUSTARÍA	3	15
GÉNERO	3	15
HABRÍAN	3	15
HERIDO	3	15
HIPÓTESIS	3	15
HUELLAS	3	15
HUIR	3	15

word	category	token
LOS	11	59
EN	11	60
A	12	61
EL	12	62
QUE	12	63
LA	13	64
Y	13	65
DE	14	66

Fig. 3. An example of categories and instances.

In the third column we find the tokens. In every category there may be one or more tokens. In category 1, for instance all the words illustrated ("ESTIRABAN",..., "ESTRAÑO") map into token 1 ("T3"). On the other hand, category 12 consists of three tokens and category 14 consists of only one token. Any word in the text may be either matched up with a token (if it is found in the catalog) or is assigned a special "null" token. There is one possible null token for every category. Under this classification, the sentence "*Tomaré el autobús a las dos y llegaré por la tarde*" would become the set of tokens <T6> <T62> <T21> <T61> <T54> <T42> <T65> <T4> <T56> <T64> <T34>. However, this simple approach is not the one we took.

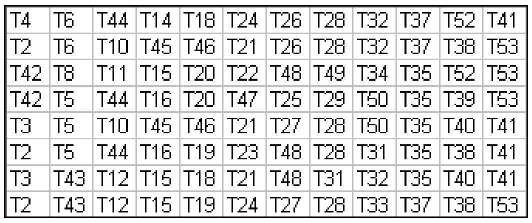
Rather, we define a zentence as a collection of 10 tokens. Our program reads words from a text and would "like" to find one representative for every category. This is attempted by reading up to 26 words or when all the 12 categories are filled-in. The number "26" was calculated so that the average number of unfulfilled categories is approximately 15%. If, after having read 26 words a category is still not yet filled-in, a null token is assigned to the empty position of the zentence. If a word is a representative of a category which has already been filled-in, that word is not considered and a new one is read. A zentence has the structure illustrated in Table 1.

Zentence No.	Token No.									
1	1	2	3	4	5	6	7	8	9	10
2	1	2	3	4	5	6	7	8	9	10

Table 1. Structure of a zentence.

2.3 Obtaining the Tokenized Data Base

As described, not all zentences include instances of all categories and, therefore, a special token ("nul_i") may appear in the *i-th* position of the zentence. That is, if no representative of category 1 appears in the zentence, it will be filled-in with the first null; if no representative of category 2 appears it will be filled-in with the second null and so on. In other words, there are up to *(c+1)t* symbols present in the database. In the example, therefore, there are up to 60 different categorical instances present in the database. This is illustrated in Figure 4.



T4	T6	T44	T14	T18	T24	T26	T28	T32	T37	T52	T41
T2	T6	T10	T45	T46	T21	T26	T28	T32	T37	T38	T53
T42	T8	T11	T15	T20	T22	T48	T49	T34	T35	T52	T53
T42	T5	T44	T16	T20	T47	T25	T29	T50	T35	T39	T53
T3	T5	T10	T45	T46	T21	T27	T28	T50	T35	T40	T41
T2	T5	T44	T16	T19	T23	T48	T28	T31	T35	T38	T41
T3	T43	T12	T15	T18	T21	T48	T31	T32	T35	T40	T41
T2	T43	T12	T15	T19	T24	T27	T28	T33	T37	T38	T53

Fig. 4. A segment of a tokenized data base.

Once *U* is determined and categorized any given selected text in Spanish may be mapped into a relational data base formed of zentences. We selected three texts by García Márquez and three by Julio Cortázar and tokenized them accordingly. These texts were then encoded by CENG and finally clustered using Self-Organizing Maps. The resulting clusters were labeled and then tested for similarity. Clusters whose tuples are similarly labeled indicate the same author; different authors otherwise.

3. Coding and Clustering the Tokenized Data Base

Once the tokenized data base has been obtained we proceed to encode the attributes, which correspond to the tokens determined as above. These tokenized database is one in which all attributes are categorical. The idea is to find a set of codes (one for each different instance of all categories) such that the structures present in the non-numerical data set are preserved when every instance of every category is replaced by its numerical counterpart. CENG is an algorithm based on the premise that patterns are preserved if all attributes are adequately expressed as a function of all others <u>and</u> that this is true for all attributes simultaneously. Let us assume that we have a hypothetical *perfect* set of pattern preserving codes. Assume, also, that there are n attributes and p tuples. Given such *perfect* set it would be, in principle, possible to express attribute i as a function of the remaining $n-1$ with high accuracy since this *perfect* code set will lend itself to a close approximation. Therefore, the first problem to solve, in this approach, is to find $f_1 = f(x_2 \ldots, x_n)$: a multivariate function with no predefined model for f_1; that is to say, without restricting the form of the approximant. In this regard Cybenko [13] showed that a 3-layered perceptron network (MLP) is able to approximate arbitrarily closely a function from a known data set. This theorem is not constructive and one has to design the MLP adequately. The issues involved in its design are discussed and solved in [14]. To complete the argument regarding the preservation of patterns we must find the whole collection $f_2, \ldots, f_n$ where $f_i = f(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ such that the approximation error should be the smallest <u>for all attributes</u> given the same fixed codes. This multi-objective optimization problem [since minimizing the approximation error for f_i may be in conflict with minimizing the corresponding error for f_j $(i \neq j)$] is the second problem to solve. On the other hand, it is clear that we do not know the values of what we have called the *perfect* code. These issues are solved in CENG by using EGA and MLPs.
In every generation *PS* individuals are delivered to a MLP for evaluation. Every individual consists of a set of possible codes which are to be assigned to every instance in the database. An individual for the database illustrated in Figure 5a is shown in Figure 5b. Figure 5c illustrates the database resulting from replacing the instances of MD with the codes of Figure 5b. The numbers illustrated in Figure 5b are actually encoded in binary with 22 bits each. Numerical variables are mapped into [0,1) so that all numbers lie in the same range.

Age	Place of Birth	Years of Study	Race	Sex	Salary
55	North	9	White	M	2,932.49
62	North	7	Asian	F	23,453.37
57	South	24	Indian	F	1,628.61
56	Center	18	White	M	4,069.62
49	South	22	Indian	F	3,650.23

Fig. 5a. A mixed database (MD)

North	Center	South	White	Asian	Indian	Other	M	F
0.0002	0.2691	0.2209	0.8304	0.0668	0.4084	0.7472	0.1332	0.1283

Fig 5b. Instances of MD and possible codes

Age	Place of Birth	Years of Study	Race	Sex	Salary
0.4928	0.0002	0.2000	0.8304	0.1332	0.0226
0.5942	0.0002	0.1200	0.0668	0.1283	0.1896
0.5217	0.2209	0.8000	0.4084	0.1283	0.0120
0.5072	0.2691	0.5600	0.8304	0.1332	0.0318
0.4058	0.2209	0.7200	0.4084	0.1283	0.0284
0.0870	0.0002	0.8400	0.0668	0.1332	0.2306

Fig 5c. Numerical database (ND) with codes from Figure 2b

Every (binary string) individual of EGA is transformed to a decimal number and its codes are inserted into MD, which now becomes a candidate numerical data base. A MLP (whose architecture is determined as per [14]) is trained (for 1000 epochs with backpropagation [15]) using the first attribute as the output and the remaining $n-1$ as inputs to the MLP. The associated absolute error is calculated and temporarily stored. This process is repeated n-1 more times using every attribute as the output of a MLP and the remaining n-1 as its inputs. The largest of the stored errors is returned to EGA as the fitness of the individual. EGA runs for 500 generations thus minimizing the maximum absolute error. This strategy guarantees that the resulting set of codes corresponds to the best global behavior. That is, the final set of codes encompasses the best combinations of the f_i's minimizing the approximation error and the multi-objective optimization is solved within the practical limits of the EGA. Additionally it delivers a practical approximation of the *perfect* set: the one preserving the patterns embedded in the data base.

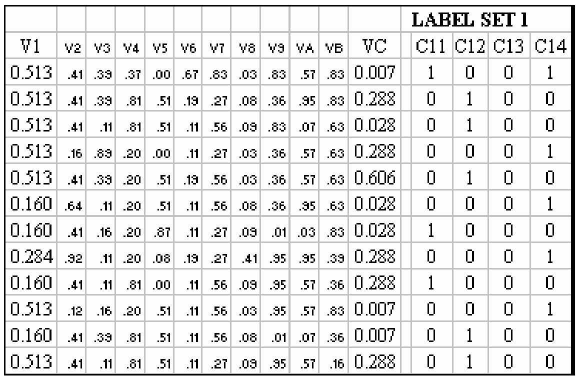
Since both initial values of the weights during the training phase of the MLPs and the initial population of EGA are random any two runs of CENG, in general, will result in different sets of codes, say S1 and S2. This fact allows us to verify that, as postulated, patterns will be preserved. This is done by applying a clustering algorithm which yields an assignment of every tuple to one of *m* clusters. Under the assumption of pattern preservation, clustering with S1 and S2 should yield analogous clusters. This is, indeed, the fact, as was shown in [8].

4. Identification of Cluster Matching in Labeled Databases

The correct identification of analogous clusters is compulsory if, as intended, we are to determine whether two texts correspond (or not) to the same author. Texts T1A and T2A both authored by A should correspond to similar clusters whereas texts T1A and T1B (authored, respectively, by A and B) should correspond to different clusters. To test the purported cluster similarity poses the technical problem we describe in what follows.

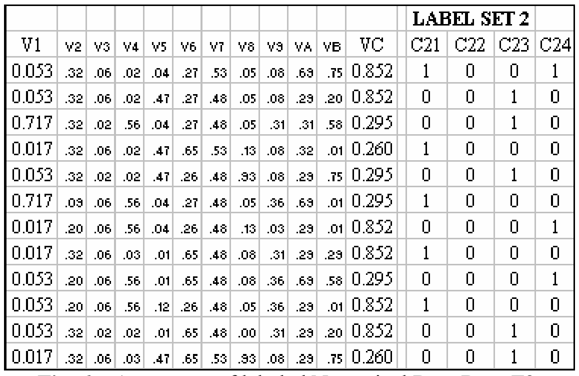
4.1 The problem of cluster matching

Assume that tables T1 and T2 consisting of attributes V1,...,VC have been classified into 4 clusters and labeled as illustrated in Figures 6a and 6b. This labeling convention is convenient since one may easily count the number of matches between two clusters. However, clustering algorithms do not necessarily yield the same order of the label columns. For example, in Figure 7 we have compared column C11 to C21, on the one hand and C11 to C24 on the other. The first choice yields a count of 6 matches leading us to the conclusion that sets C11 and C21 do not match and that, therefore, T1 and T2 do not share the same clusters.
The second choice, however, yields the full 12 matches. Therefore, in this instance one must conclude that column C11 (from set 1) actually corresponds to column C24 (from set 2). Accordingly, we should also conclude that T1 and T2 correspond to the same set for cluster 1. The correct pairing has to be achieved in similar fashion for all clusters.



												LABEL SET 1			
V1	V2	V3	V4	V5	V6	V7	V8	V9	VA	VB	VC	C11	C12	C13	C14
0.513	.41	.39	.37	.00	.67	.83	.03	.83	.57	.83	0.007	1	0	0	1
0.513	.41	.39	.81	.51	.19	.27	.08	.36	.95	.83	0.288	0	1	0	0
0.513	.41	.11	.81	.51	.11	.56	.09	.83	.07	.63	0.028	0	1	0	0
0.513	.16	.89	.20	.00	.11	.27	.03	.36	.57	.63	0.288	0	0	0	1
0.513	.41	.39	.20	.51	.19	.56	.03	.36	.57	.63	0.606	0	1	0	0
0.160	.64	.11	.20	.51	.11	.56	.08	.36	.95	.63	0.028	0	0	0	1
0.160	.41	.16	.20	.87	.11	.27	.09	.01	.03	.83	0.028	1	0	0	0
0.284	.92	.11	.20	.08	.19	.27	.41	.95	.95	.39	0.288	0	0	0	1
0.160	.41	.11	.81	.00	.11	.56	.09	.95	.57	.36	0.288	1	0	0	0
0.513	.12	.16	.20	.51	.11	.56	.03	.95	.57	.83	0.007	0	0	0	1
0.160	.41	.39	.81	.51	.11	.56	.08	.01	.07	.36	0.007	0	1	0	0
0.513	.41	.11	.81	.51	.11	.27	.09	.95	.57	.16	0.288	0	1	0	0

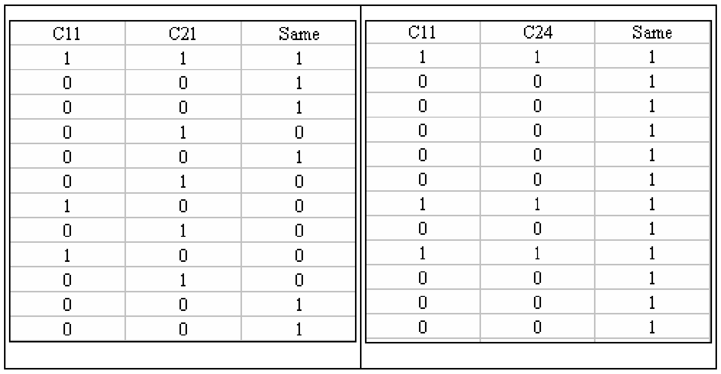
Fig. 6a. A segment of labeled Numerical Data Base T1.



												LABEL SET 2			
V1	V2	V3	V4	V5	V6	V7	V8	V9	VA	VB	VC	C21	C22	C23	C24
0.053	.32	.06	.02	.04	.27	.53	.05	.08	.69	.75	0.852	1	0	0	1
0.053	.32	.06	.02	.47	.27	.48	.05	.08	.29	.20	0.852	0	0	1	0
0.717	.32	.02	.56	.04	.27	.48	.05	.31	.31	.58	0.295	0	0	1	0
0.017	.32	.06	.02	.47	.65	.53	.13	.08	.32	.01	0.260	1	0	0	0
0.053	.32	.02	.02	.47	.26	.48	.93	.08	.29	.75	0.295	0	0	1	0
0.717	.09	.06	.56	.04	.27	.48	.05	.36	.69	.01	0.295	1	0	0	0
0.017	.20	.06	.56	.04	.26	.48	.13	.03	.29	.01	0.852	0	0	0	1
0.017	.32	.06	.03	.01	.65	.48	.08	.31	.29	.29	0.852	1	0	0	0
0.053	.20	.06	.56	.01	.65	.48	.08	.36	.69	.58	0.295	0	0	0	1
0.053	.20	.06	.56	.12	.26	.48	.05	.36	.29	.01	0.852	1	0	0	0
0.053	.32	.02	.02	.01	.65	.48	.00	.31	.29	.20	0.852	0	0	1	0
0.017	.32	.06	.03	.47	.65	.53	.93	.08	.29	.75	0.260	0	0	1	0

Fig. 6a. A segment of labeled Numerical Data Base T2.

If there are m clusters and p tuples there are m^p possible combinations of valid labeling sets. We need to investigate which of these does actually correspond to the proper matching of the m clusters in T1 with those of T2. Only then we may compare T1 and T2 and determine their similarity. To achieve this identification we designed the following algorithm.



C11	C21	Same
1	1	1
0	0	1
0	0	1
0	1	0
0	0	1
0	1	0
1	0	0
0	1	0
1	0	0
0	1	0
0	0	1
0	0	1

C11	C24	Same
1	1	1
0	0	1
0	0	1
0	0	1
0	0	1
0	0	1
1	1	1
0	0	1
1	1	1
0	0	1
0	0	1
0	0	1

Fig. 7. Similarity for different choices of cluster columns

4.2 An algorithm to optimize cluster matching

```
1. Create a matching table "MT" of dimensions $m \times m$.
   Make MT($i,j$) $\leftarrow$ 0 for all $i, j$.
2. For $i \leftarrow 1$ to $m$
       For $j \leftarrow 1$ to $m$
           If column $i$ = column $j$
               MT($i,j$) $\leftarrow MT(i,j) + 1$
           endif
       endfor
   endfor
```

MT(i,j) will contain the number of matches between cluster i of table T1 and cluster j of table T2.

```
3. Create a table "Scores" of dimension $Q$   ($Q$ >>0).
4. For $i \leftarrow 1$ to $Q$
       4.1. Set a random valid sequence $S_i$ of $m$ possible matching sequences between
            the clusters of T1 and those of T2.
       4.2. Find the number of matches $M_i$ between T1 and T2 from table MT as per $S_i$.
       4.3. Make Scores(i) $\leftarrow M_i$.
   endfor
5. $I \leftarrow$ index of max(Scores($i$)) for all $i$.
6. Select $S_I$. This is the matching set which maximizes the number of coincidences
   between the clusters of T1 and T2.
```

The core of the algorithm lies in step 4.1 where the valid matching sequences are determined. This algorithm will find the sequence which maximizes the number of matches between the clusters of T1 and T2 with high probability provided Q is large enough. In our experiments we made Q =1000. Given the large number of possible pairings between the clusters of T1 and T2 the algorithm is a practical way to select which cluster of T1 should be paired with which cluster of T2.

5. Experimental Authorship Identification

At this point we are in a position which allows us to test the initial hypothesis of authorship identification. As already stated, we selected 3 texts from Gabriel García Márquez (GM) and 3 from Julio Cortázar (JC). These were, consecutively, tokenized, CENG-encoded, clustered with SOMs and labeled. Previous analysis led us to the conclusion that there were 4 clusters, as may be seen from the graph in Figure 8.

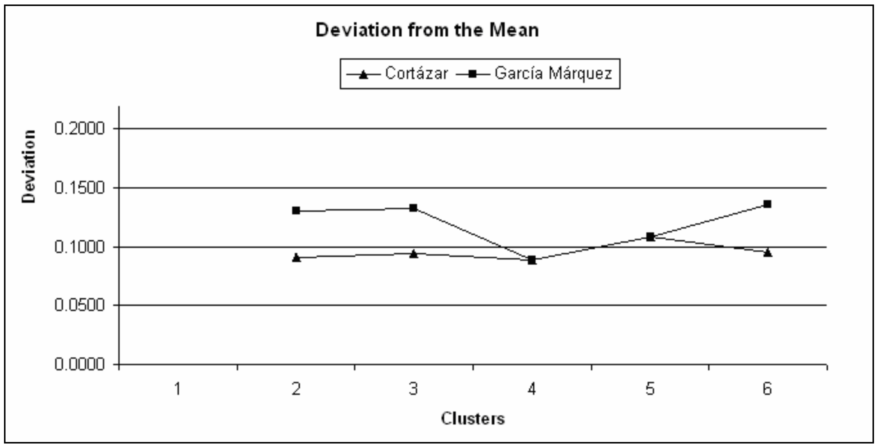
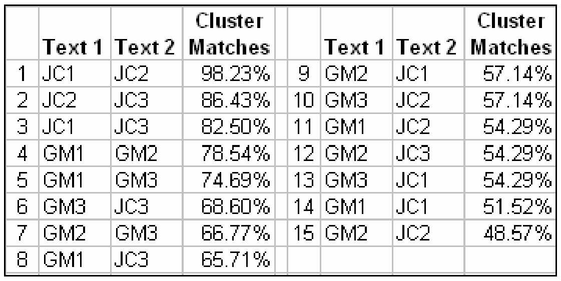


Fig. 8. Standard deviation for training errors

In this graph we display the results of having trained texts from GM and JC for up to 6 clusters with SOMs. The maximum errors relative to the mean were smallest for 4 clusters and the calculated standard deviation with a 0.05 *p-value* [16] was, likewise, smallest for the same number.
The texts we selected were of roughly the same size: ≈16,000 words. They were then tokenized and CENG-encoded. The resulting data bases were clustered and labeled. Next the cluster matching was performed and adjusted when needed. We then proceeded to obtain a matrix of coincidences for the 15 possible combinations of the 6 texts. These are shown in Table 2. GM_i denotes the *i-th* text by García Márquez; likewise JC_i denotes the texts by Cortázar.



	Text 1	Text 2	Cluster Matches		Text 1	Text 2	Cluster Matches
1	JC1	JC2	98.23%	9	GM2	JC1	57.14%
2	JC2	JC3	86.43%	10	GM3	JC2	57.14%
3	JC1	JC3	82.50%	11	GM1	JC2	54.29%
4	GM1	GM2	78.54%	12	GM2	JC3	54.29%
5	GM1	GM3	74.69%	13	GM3	JC1	54.29%
6	GM3	JC3	68.60%	14	GM1	JC1	51.52%
7	GM2	GM3	66.77%	15	GM2	JC2	48.57%
8	GM1	JC3	65.71%				

Table 2. Comparison of clusters obtained

The texts pairings were ordered according to the percentage of matches we obtained. We observed the following behavior:

- All matching text matches were higher when the authors were the same, with the exception of item 6, where the texts by GM vs JC had higher matches than expected.
- The correct assessment of authorship matches for the first 5 couples remains very close or above 75% Therefore, a matching percentage of 75% appears to be sufficient to ascertain similar authorship.
- There appears to be no possible definitive conclusions of the purported authorship in the borderline percentages for items 6-8.
- Matching percentages below 60% seem to imply negative authorship for the analyzed couples.
- The identification percentage falls smoothly so that there is not a clear cut threshold dividing correctly assessed authorship from the alternative.

6. Conclusions

We have described a method to identify the authorship of selected Spanish texts which is not based on linguistic considerations. Rather, it relies on the identification and preservation of the patterns embedded in the texts by the intelligent encoding of the data. This method is computationally intensive, requiring the training of a large set of candidate codes and, therefore, CPU consuming. Execution time may be improved by the use of parallel computation from which genetic algorithms will benefit considerably. On the other hand, there are several parameters which were heuristically determined and have to be further explored. For instance, the size of the zentences is arbitrary. Likewise, the fact of having defined the number of categories and the corresponding instances as they were is arbitrary and we expect that a more elaborate selection of these (after systematic experimental tests) may improve the algorithm significantly. Furthermore, the selected texts and their lengths may affect the efficiency of the algorithm one way or the other. Finally, the results, which seem to be promising, are only valid, in general, if we assume that the method will behave similarly if the restricted number of authors we selected were to be expanded. Even if it is not reasonable to assume that the reported results should be much different for a

different selection of authors it could be that such is the case. Much experimental work remains to be done.
At any rate we believe this to be a promising and novel alternative. Particularly in view of the fact, that as pointed out in the introduction, it may be applied to any kind of unstructured data. We expect to report on the application of our method to more general and non-textual data in the near future.

References

[1] http://news.netcraft.com/archives/2014/04/02/april-2014-web-server-survey.html, [last retrieved 06-18-2015]
[2] Odlyzko, A., & Tilly, B. (2005). A refutation of Metcalfe's Law and a better estimate for the value of networks and network interconnections. Manuscript, March, 2, 2005.
[3] http://www-03.ibm.com/press/us/en/pressrelease/46205.wss, [last retrieved 06-18-2015].
[4] Tan, A. H. (1999, April). Text mining: The state of the art and the challenges. In Proceedings of the PAKDD 1999 Workshop on Knowledge Disocovery from Advanced Databases (Vol. 8, p. 65).
[5] Pachet, F., Westermann, G., & Laigre, D. (2001, November). Musical data mining for electronic music distribution. In Web Delivering of Music, 2001. Proceedings. First International Conference on (pp. 101-106). IEEE.
[6] Lei-da Chen, T. S., & Frolick, M. N. (2000). Data mining methods, applications, and tools.
[7] Feldman, R., & Sanger, J. (2007). The text mining handbook: advanced approaches in analyzing unstructured data. Cambridge University Press.
[8] Kuri-Morales, A., (2015, June). Categorical Encoding with Neural Networks and Genetic Algorithms. In Proceedings of AICT'15 (Applications of Computer and Computer Theory), Salerno, Italy, (167-175). WSEAS.
[9] Kuri-Morales, A., Aldana-Bobadilla, E. (2013). The best genetic algorithm I. Advances in Soft Computing and Its Applications. Springer Berlin Heidelberg, 16-29.
[10] Bezdek, J. C., Ehrlich, R., & Full, W. (1984). FCM: The fuzzy c-means clustering algorithm. Computers & Geosciences, 10(2), 191-203.
[11] Ritter, H., Martinetz, T., & Schulten, K. (1992). Neural computation and self-organizing maps. An introduction.
[12] Aldana-Bobadilla, E., & Kuri-Morales, A. (2015). A Clustering Method Based on the Maximum Entropy Principle. Entropy, 17(1), 151-180.
[13] Cybenko, G. (1989) Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems 2.4 303-314.
[14] Kuri-Morales, A., (2014). The Best Neural Network Architecture, Nature Inspired Computation and Machine Learning, LNAI 8857, Springer, 72-84.
[15] Hecht-Nielsen, R. (1989, June). Theory of the backpropagation neural network. In Neural Networks, 1989. IJCNN., International Joint Conference on (pp. 593-605). IEEE.
[16] Westfall, P. H., & Young, S. S. (1993). Resampling-based multiple testing: Examples and methods for p-value adjustment (Vol. 279). John Wiley & Sons.