The Rose-Gurewitz-Fox approach applied for patents classification

Ioan Bogdan Hodrea [*] Radu Ioan Boţ [†] Gert Wanka [‡]

Abstract. We used the so-called deterministic annealing algorithm due to Rose and Gurewitz by the classification of patent documents. A C++ program based on this algorithm was run first on some artificially constructed data and the results were good. Then we tested it on data sets obtained from some already classified patents. The conclusion we reached is that this algorithm provides an alternative classification to the one used in the US Patent Classification System.

Key Words. Unsupervised Classification, Deterministic Annealing, Data Mining, Patent Classification

1 Introduction

Segmentation of large data sets into smaller homogeneous subsets that can be easily managed, separately modelled and analyzed is a fundamental operation in data mining. Clustering is a very popular approach used to implement this

[*]Faculty of Mathematics, Chemnitz University of Technology, D-09107 Chemnitz, Germany, e-mail: hio@mathematik.tu-chemnitz.de

[†]Faculty of Mathematics, Chemnitz University of Technology, D-09107 Chemnitz, Germany, e-mail: radu.bot@mathematik.tu-chemnitz.de.

[‡]Corresponding author. Faculty of Mathematics, Chemnitz University of Technology, D-09107 Chemnitz, Germany, e-mail: gert.wanka@mathematik.tu-chemnitz.de

operation, especially when we deal with data about whose internal structure little or no prior information is available.

Consider a data set $X = \{x_1, ..., x_N\}$ and a measure of distance or similarity d. The goal of most clustering algorithms is to assign each data point to a cluster. The arising partition reflects somehow the internal structure of the data and identifies "natural" classes and hierarchies contained within.

Let us consider $Y = \{y_1, ..., y_{N_C}\}$ a set of codevectors (being the centers of the clusters). By the center of a cluster we understand a vector which characterizes the elements of the cluster relative to a certain distance measure. For instance this could be the center of the mass or the point which minimizes the sum of the distances to the elements of the cluster.

For an arbitrary vector x_i in the above data set and a codevector y_j, we denote by $d(x_i, y_j)$ the distortion measure between them. The global distortion is given by the formula

$$D = \sum_{\substack{i=1,...,N,\\ j=1,...,N_C}} p(x_i, y_j) d(x_i, y_j), \tag{1}$$

where $p(x_i, y_j)$ is the probability that the vector x_i belongs to the cluster represented by the codevector y_j. The goal of clustering is to minimize the global distortion. Being a function of the codevectors y_j and the associated probabilities $p(x_i, y_j)$, D is usually nonconvex and has several local minima. However a global minimum is quite impossible to be determined in practice (cf. [7]).

During the last few decades many new and interesting books and papers have enriched the literature on clustering. Among the ones that made themselves useful to us, let us mention [3], [4], [11], [12], [14], [15]. Various algorithms have been proposed within these works, their origins being recognizable in graph theory, pattern recognition, communication theory, statistical mechanics and some

other fields. A common problem encountered in the papers presenting applications of these algorithms resides in the fact that different methods may return more or less similar results when using the same inputs. These differences are somehow expected because of the various ways these algorithms treat the clustering problem. Although many of the algorithms used in clustering are capable to lead to optimal solutions a major drawback comes from the fact that external parameters influence the results.

Some algorithms (like K-mean and K-median) require the number of clusters to be given in the beginning. Therefore, we do not use them for our tests.

Trying to avoid the above mentioned difficulties and being motivated by good results and reviews (see for example [1]) in the literature, we have decided to construct and use a program based on the algorithm introduced by Rose in [11] to clusterize some classes of patents.

To obtain the global minima of the distortion, Rose et al. ([11], [12], [14]) used some methods inspired from annealing processes in physical chemistry. What is specific about this algorithm is its property to avoid getting trapped into local minima. Also the fact that the number of created clusters is influenced by a parameter modified inside the algorithm (and not given in the beginning) is an important and useful property.

The present paper is organized as follows. The next section contains a brief description of an algorithm based on the deterministic annealing. A complete description of the method including some necessary proofs is available in [11], [12], [14]. The third part presents the results we have obtained by running the mentioned program on some artificial data sets we deliberately constructed. As these results were promising, we have dedicated the fourth section to running the program on some real data obtained from the US Patent Classification System.

A small conclusive section ends our paper.

2 Description of the method

Since its introduction due to Rose et al. in [14], deterministic annealing has proven to be a powerful tool in data mining. Various authors have developed different versions of the method, treating a large variety of problems, from speech recognition and signal processing to unsupervised texture segmentation.

In the following we will treat the expected distortion in (1) in a probabilistic way, as follows

$$D = \sum_{\substack{i=1,...,N,\\ j=1,...,N_C}} p(x_i, y_j) d(x_i, y_j) = \sum_{i=1,...,N} p(x_i) \sum_{j=1,...,N_C} p(y_j|x_i) d(x_i, y_j), \quad (2)$$

where we denote by $p(x_i, y_j)$ the joint probability distribution and by $p(y_j|x_i)$ the association probability relating the vector x_i and the codevector y_j. For $j = 1, ..., N_C$ we denote by $p(y_j)$ the probability distribution induced on the codevectors using the following encoding rule $p(y_j) = \sum_{i=1,...,N} p(x_i) p(y_j|x_i)$.

The global distortion has to be minimized as a function of y_j and $p(x_i, y_j)$. In order to find a solution to this optimization problem the objective function D is regularized through a penalty term having the form of the Shannon entropy

$$H(X, Y) = -\sum_{i=1,...,N} \sum_{j=1,...,N_C} p(x_i, y_j) \ln(p(x_i, y_j)).$$

The function we have to minimize now is

$$F = D - TH,$$

where D is given in (2) and T is a positive regularization parameter. When T is very large the new problem turns into the entropy maximization, but as T tends to zero the solution approaches the one of the initial problem, which is actually our aim.

Thus, in order to find the minimal value of D, we have to minimize F with respect to the codevectors $y_j \in Y$ and the associated probabilities $p(y_j|x_i)$. We start with a high value of T and the minimum is tracked while lowering T. It can be proved that the central iteration which allows us to calculate the minimum consists of the following two steps:

(i) fix the codevectors $y_j \in Y$ and calculate the associated probabilities $p(y_j|x_i)$ using the formula

$$p(y_j|x_i) = \frac{p(y_j)e^{-\frac{d(x_i,y_j)}{T}}}{\sum\limits_{j=1,...,N_C} p(y_j)e^{-\frac{d(x_i,y_j)}{T}}} \tag{3}$$

(ii) fix the associated probabilities $p(y_j|x_i)$ and calculate the codevectors $y_j \in Y$ by solving

$$\sum_{i=1,...,N} p(y_j|x_i)\frac{d}{dy_j}d(x_i,y_j) = 0 \tag{4}$$

for each $j = 1, ..., N_C$.

The complete procedure, including some proofs, is described in [11]. We would like also to mention that as T tends to zero a hard clustering solution is obtained, i.e. each vector x_i is associated to a single codevector y_j and the associated probability $p(y_j|x_i)$ tends to 1.

Like in physics, where the algorithm is rooted, let us further call the parameter T temperature and consider the vectors as points of a certain system. As the temperature is lowered, it has been proven (see [11]) that the system undergoes

a sequence of phase transitions, which consists of natural cluster splits, where the clustering model grows in size (number of clusters). Concerning the cluster splits, we have the following statement.

Theorem. (cf. [11]) *For the squared error distortion measure, a cluster centered at codevector y_j undergoes a splitting phase transition when the temperature reaches the critical value $T = 2\lambda_{\max}$, where the $\lambda_{\max}$ is the largest eigenvalue of the covariance matrix $C_{x|y_j} = \sum_{i=1,...,N} p(x_i|y_j)(x_i - y_j)(x_i - y_j)^T$.*

A brief description of the algorithm we used to develop the program we ran tests with follows. It has been introduced by Rose in [11], while some earlier versions of it are available in [12], [14].

First we set a minimal temperature $T_{\min}$, a maximal number of clusters $K_{\max}$ and we consider as distance measure the squared error distortion measure. When the temperature becomes $T_{\min}$ (or lower) or the number of clusters reaches $K_{\max}$ (or more), the algorithm stops. The initializing step consists in taking the number of clusters $K = 1$, i.e. all vectors to belong to the same cluster centered in $y_1 = \sum_{i=1,...,N} x_i p(x_i)$, where $p(x_i) = \frac{1}{N}$, $i = 1, ..., N$, and introducing $p(y_1) = 1$. The initial temperature T is chosen such that $T > 2\lambda_{\max}$, where λ_{max} is the largest eigenvalue of the covariance matrix $C_{x|y_1}$. Let us suppose that we have already created K clusters. For calculating the codevectors y_j, $j = 1, ..., K$, we use the formulas $(3) - (4)$, which lead us to

$$y_j = \frac{\sum_{i=1,...,N} x_i p(x_i) p(y_j|x_i)}{p(y_j)}, j = 1, ..., K,$$

where

$$p(y_j|x_i) = \frac{p(y_j)e^{-\frac{(y_j-x_i)^2}{T}}}{\sum\limits_{j=1,...,K} p(y_j)e^{-\frac{(y_j-x_i)^2}{T}}}, i = 1, ..., N, j = 1, ..., K$$

and

$$p(y_j) = \sum_{i=1,...,N} p(x_i)p(y_j|x_i), j = 1, ..., K.$$

Therefore we generate a convergent sequence of vectors until the distance between two consecutive terms is less than or equal to a given error (this means that the distance between the "old" codevector y_j and the "new" codevector y_j must be less than or equal to the given error for each $j = 1, ..., K$). When the new codevectors are obtained we check the temperature. If the temperature of the system is smaller than or equal to $T_{\min}$ the algorithm stops. Otherwise we perform a cooling step, T being replaced by αT, where $\alpha \in (0, 1)$. If $K < K_{\max}$, for each cluster $i = 1, ..., K$ we calculate its associated temperature (i.e. the double of the largest eigenvalue of the covariance matrix $C_{x|y_i}$) and we compare it to T. If the temperature of one cluster $j \in \{1, ..., K\}$ surpasses the one of the system, a new center y_{K+1} is introduced and K grows by 1. The new value of the probability $p(y_j)$ will be half from the value $p(y_j)$ and the same value is assigned to the probability $p(y_{K+1})$. For the resulting centers we generate again the convergent sequences described above and we repeat the procedure until either $T \leq T_{\min}$ or $K \geq K_{\max}$.

Further we give the pseudo-code of the algorithm described above.

N = number of documents, W = number of words,

$X = \{x_1, ..., x_N\}$, $x_k = (x_k^1, ..., x_k^W), k = 1, ..., N$

(1) Stop Conditions: number of codevectors K_{max}, minimum temperature T_{min}.

(2) Initialize: $\varepsilon > 0$ small enough, $\alpha(0 < \alpha < 1)$, $K = 1$, $y_1 = \frac{1}{N} \sum_{k=1,...,N} x_k$, $p(y_1) = 1$, $T > 2\lambda_{max}(C_{x|y_1})$ and $p(x_k) = \frac{1}{N}, \forall k = 1, ..., N$.

(3) For all $i = 1, ..., K$ calculate

$$p(y_i|x_k) = \frac{p(y_i)e^{-((x_k - y_i)^2/T)}}{\sum_{j=1,...,K} p(y_j)e^{-((x_k - y_j)^2/T)}}, k = 1, ..., N,$$

$$p(y_i) = \frac{1}{N} \sum_{k=1,...,N} p(y_i|x_k)$$

and

$$y_i' = \frac{1}{N} \frac{\sum_{k=1,...,N} x_k p(y_i|x_k)}{p(y_i)}.$$

(4) Convergence test:

If there exist $j \in \{1, ..., K\}$ such that $d(y_j, y_j') > \varepsilon$, the convergence test is not satisfied. Let $y_i := y_i'$, $\forall i \in \{1, ..., K\}$ and go to (3).

Otherwise $y_i := y_i'$, $\forall i \in \{1, ..., K\}$.

(5) If $T \leq T_{min}$ STOP.

(6) Cooling Step: $T \leftarrow \alpha T$, $(\alpha < 1)$.

(7) If $K \geq K_{max}$ STOP.

If $K < K_{max}$, check the temperature of the clusters. If critical T is reached

for cluster j, $j \in \{1, ..., K\}$, add a new codevector $y_{K+1} = y_j + \delta$, $p(y_{K+1}) = p(y_j)/2$, $p(y_j) \leftarrow p(y_j)/2$ and increment K.

(8) Go to (3).

As a remark, let us mention that in order to calculate the covariance matrix $C_{x|y_j}$ we need to calculate the probabilities $p(x_i|y_j)$. Since inside the algorithm we calculate the probabilities $p(y_j|x_i)$, the above mentioned probabilities are straightforward to calculate using Bayes formula

$$p(x_i|y_j)p(y_j) = p(y_j|x_i)p(x_i),$$

for each $i = 1, ..., N$ and $j = 1, ..., K$.

To determine the largest eigenvalue of the covariance matrix associated to a center y we have used the so-called Power Method Algorithm ([18]). When a cluster is split its center remains as center of one of the new clusters, while the center of the other is taken initially on the principal axis of the old cluster, i.e. on the direction given by the eigenvector associated with the mentioned largest eigenvalue. The distance between the old and the new center has to be equal to 1. The tests have shown that the results obtained by this choice of the new center are considerably better than for any other option.

We have implemented a C++ program based on this algorithm. Several tests have been run on this programm using artificial, as well as real data. The inputs of the program consisted in a documents-words matrix, whose rows play the role of the vectors mentioned above, since any document may be represented as a vector of words. As output, each document is assigned to a unique cluster represented by its center.

3 Tests with artificial data

The needs to process larger and larger quantities of data have increased dramatically during the last decades. In the last few years the data sets used in the applications presented in the literature on data mining began to weight hundreds of Megabytes and even Gigabytes. This requires the algorithms used in the data mining to be able to deal with such huge amounts of information. As many algorithms introduced earlier provided good results when tested on small-sized data sets and irrelevant ones for larger quantities of information, we have built first some carefully constructed data sets in order to test the algorithm on huge amounts of data. A first purpose of these tests consisted in obtaining thousands of clusters, while in the literature their maximal number scarcely overcomes a couple of hundreds. Another followed scope is to check the liability of the algorithm when the dimension of the involved vectors surpasses several thousands, as the real data used in our tests contains smaller-dimensioned vectors.

In order to be able to verify the results we built the data set as separable as possible. Therefore we choose a dimension W and the number of clusters N_C. Each cluster will be constructed as follows. First we randomly generate a vector $y = (y^1, ..., y^W)$ of dimension W meant to be the centroid of a certain cluster. Assume that we want this cluster to contain m vectors and let $X = \{x_1, ..., x_m\}$ be their set. Each vector $x_i = (x_i^1, ..., x_i^W)$, $i = 1, ..., m$, is obtained as follows. The value of x_i^j, $j = 1, ..., W$, is obtained by the method described in [5]. This method is based on subtracting of 6 out of the sum of 12 randomly generated real numbers between 0 and 1. These m vectors are grouped by assigning to a ball of radius 6 centered in the origin. We translate these vectors around y by using the formula $x_i^j := y^j + \beta \cdot x_i^j$, $i = 1, ..., m$, $j = 1, ..., W$, where β is a real number

greater than or equal to 1. The role of this β is to variate the radius of different clusters. We perform the same operation for all of the clusters. It is easy to remark that each cluster we have generated approximates a Gauss distribution (cf. [5]). The pseudo-code of the algorithm described above follows.

(1) Initialize: m.

(2) Generate the center y

for $j = 1, ..., W$

$y^j = \text{rand}(0, 100)$;

(3) Generate the vectors

$\beta = \text{rand}(1, 20)$;

for $i = 1, ..., m$

(a) Generate the vector

for $j = 1, ..., W$

$x_i^j = 0$;

for $l = 1, ..., 12$

$x_i^j = x_i^j + \text{rand}(0, 1)$;

$x_i^j = x_i^j - 6$;

(b) Translate the vector

for $j = 1, ..., W$

$x_i^j = y^j + \beta \cdot x_i^j$;

Each vector becomes a row in the test-matrix, the position of this row being uniquely determined between 1 and the total number of vectors taking into ac-

count that the vectors corresponding to a cluster should not all be on consecutive positions.

Our purpose is to verify if the program returns as clusters' centroids the ones we have randomly generated or, if it is not the case, how far from the latter ones are they placed.

We have generated tests for $N_C = 50, 100, 500, 1000, 1500, 3000$ clusters with m taking values around 25, but not less than 20 and $W = 1000, 2000, 3000, 4000,$ 5000. For each pair (N_C, W) we ran the program described in section 2 several times, imposing N_C as maximal number of resulting clusters. Excluding several misclassifications of some "documents", the results turned out to be as expected in most of the cases.

The table bellow presents some results obtained from this experiment.

number of clusters	100	500	800	1000	1500	3000
minimal distance	4.22014	3.96729	4.76241	4.76532	4.36249	4.83134
average distance	8.598	8.30497	8.38439	8.30497	8.52534	8.31281
maximal distance	14.4589	15.1138	15.3579	14.2356	14.7822	14.4965

Table 1

Each column of Table 1 represents a test and the number of obtained clusters is written in the first row. It is important to know that the number of clusters generated by the program was equal with the number of clusters we want to obtain. We also mention that the tests were created using the same program, the only difference between them being the number of clusters we have generated, while the dimension of the vectors was $W = 1000$. For each cluster the distance between the initially generated centroid of the cluster and the centroid generated

by the program is calculated. On the second row the minimal value of these distances is written, while on the last row we write the maximal value. Further we have calculated the sum of these distances and divide it by the number of clusters. The value we have obtained is written on the third row of the table.

An interesting remark that arises from these tests concerns the centroids of the clusters. As we can see in the table, the mean values we obtain vary very little. Of course, as W takes larger values, the values of the distances increase, but the observation above remain valid.

4 Tests with real data

Encouraged by the promising results obtained on artificially constructed data we ran the same program on some real data sets. From the firma IP Century AG we have received some files containing information about several US Patents classes, namely $US331$, $US338$, $US703$, $US706$. The initial files contained text data where each line consisted of a word, a patent in which it appeared and the number of occurences of the word in the patent. We present further the way we dealt with the class $US706$. Similar steps were performed for all the other classes.

First we created a dictionary containing all the words that appeared in the patents included in the class, 33118 in this case. The list of the patents was extracted, too, containing 2255 pieces. Considering the dictionary as a 33118-dimensional vector, we projected each patent on it, i.e. the resulting vectors contain 0 on the positions representing words that do not appear in the document and the number of times the word appears in the patent otherwise. Thus we obtain a 2255×33118-dimensional matrix with non-negative integer entries. Further we removed the columns corresponding to the words contained in the

SMART stoplist available at [19], which were also erased from the dictionary. The well-known Porter stemmer ([20]) was used to further reduce the dimensions of the dictionary. Words having the same root remained represented by it in the new dictionary. The columns corresponding to the merged words in the matrix were summed together. As an example, after applying the Porter stemmer the following words "absorb", "absorbed", "absorber", "absorbing" and "absorbs" remained represented in the transformed dictionary only by the first one. In this way the dimension of the dictionary was reduced to 20934 words. Let us call the resulting documents-words matrix DW. There are many ways to build documents-words matrices (cf. [2]), some of them being in our attention for future research.

Our intention was to verify the "quality" of the obtained clusters, i.e. whether each of them corresponds to a single subclass of the considered class or not. Because the structure of the classes is more complicated than our needs, we have merged the higher order subclasses into the first order subclass they are derived from. In the class $US706$ they were 93 subclasses of multiple orders, but only 10 subclasses of the first order, which remained actually to work with.

In order to construct a documents-subclasses matrix we have assigned the entry (i, j), situated at the intersection of the row i and column j with the value 1 if document i has been a priori classified in the first order subclass j and/or in one of its subclasses and 0 otherwise. Let us call the resulting documents-subclasses matrix DC.

In order to avoid overusing the memory resources of the computer, we have run the program on some partial data truncated from the above obtained matrices DW and DC. For several tests we have considered only the documents in 3 or 4 subclasses. The columns of DW corresponding to the words not contained

in the documents in the chosen subclasses were removed, as well as the rows corresponding to the documents not included. A similar treatment applies for the matrix DC. The same rows as above are eliminated and the only columns remaining correspond to the chosen subclasses. The same steps were performed for the other three classes, too. Let us mention that in the tests for the classes $US331$, $US338$, $US703$, which contain 60, 46, and respectively 6 first order subclasses we have used more than 4, sometimes even 11 subclasses.

We ran the program on the transformed documents-words matrix until the number of clusters we obtain was equal to the number of the columns of the new documents-subclasses matrix. We expected that each cluster should correspond to a single subclass. However, the results did not meet our expectations, the tables below sustaining this conclusion.

Each table corresponds to a test. The tables $2(a)$ and $2(b)$ show some results concerning the class $US706$, the tables $3(a)$ and $3(b)$ are for the class $US703$, $4(a)$ and $4(b)$ refer to $US338$ and the tables $5(a)$ and $5(b)$ regard the class $US331$. The columns of the tables represent the considered subclasses, while each row represents a cluster. For example, the element situated on row 3 and column 2 in Table $2(a)$ represents the number of documents from cluster III which belong to the subclass $US706/011$.

For each choice of subclasses we have run the program for different values of the parameters, the only constant remaining the number of clusters which had to coincide to the number of subclasses. More than 5 test were performed each time, but sometimes we made even 20. To avoid an unnecessary overloading of the paper we have selected only the results of one test each time.

cluster/class US706	001	011	062	902
I	8	8	2	22
II	3	-	1	1
III	-	2	2	2
IV	1	3	2	4

Table 2(a) (61 patents, 7320 words)

cluster/class US706	001	011	062	902
I	8	8	2	21
II	3	-	1	2
III	-	2	-	-
IV	1	3	4	6

Table 2$'$(a) (61 patents, 7320 words)

cluster/class US706	010	011	900
I	9	9	8
II	-	3	-
III	5	3	4

Table 2(b) (41 patents, 5329 words)

cluster/class US703	001	002	003
I	14	19	10
II	2	9	2
III	-	-	1

Table 3(a) (57 patents, 6901 words)

cluster/class US703	0061	013	023
I	28	22	22
II	3	9	7
III	5	6	7

Table 3(b) (109 patents, 9266 words)

cluster/class US338	002	007	049	202	221	279
I	3	-	-	-	-	-
II	45	7	4	16	6	7
III	1	-	-	1	-	-
IV	-	-	-	1	-	-
V	2	1	-	1	-	-
VI	9	-	1	1	-	1

Table 4(a) (107 patents, 4154 words)

cluster/class US338	050	051	053	202	210	223	277	283	295	296	333
I	-	-	-	-	-	-	-	1	1	-	1
II	-	-	1	1	3	-	1	2	-	-	-
III	-	-	-	-	-	-	1	-	-	-	-
IV	2	-	-	-	1	-	1	1	1	1	-
V	-	-	1	1	3	-	-	-	-	-	-
VI	-	-	-	-	-	-	-	-	-	1	-
VII	1	4	2	17	12	4	1	6	2	4	4
VIII	-	-	-	-	-	-	1	-	1	-	1
IX	1	1	2	1	2	1	1	-	1	-	1
X	1	1	-	-	3	2	-	-	-	-	-
XI	1	-	1	-	-	-	-	1	-	-	-

Table 4(b) (106 patents, 4144 words)

cluster/class US331	002	010	030	045	068
I	20	40	2	10	13
II	17	20	5	4	12
III	-	2	-	-	-
IV	2	10	1	1	4
V	1	-	-	-	-

Table 5(a) (164 patents, 4407 words)

cluster/class US331	003	008	014	034	046	065	094.1	143	172	185
I	6	6	9	9	5	7	-	4	8	3
II	1	-	1	-	1	-	1	-	-	1
III	-	-	1	1	1	-	-	-	1	-
IV	-	-	-	1	-	2	-	-	-	-
V	-	3	1	3	1	3	-	3	1	2
VI	1	3	8	6	7	6	5	3	-	2
VII	-	3	2	3	2	6	-	-	2	2
VIII	8	13	25	25	22	20	12	14	16	19
IX	3	4	4	7	10	4	5	3	3	2
X	4	2	1	1	5	6	-	1	-	-

Table 5(b) (386 patents, 6308 words)

As we can see in the above tables there is no complete coverage of a subclass by a single cluster that contains no documents from the other subclasses. An interesting example is provided by the Table 4(a). The cluster I contains exactly 3 documents, all belonging to the subclass $US338/002$. In the same time in the cluster II there are 45 documents previously classified in the same subclass, $US338/002$. The other documents from this subclass can be found in the clusters III, V and VI, the last one containing not so few, 9. On the other hand, all the documents in the subclass $US338/221$ were classified by the program in the cluster II, which contains documents from all the subclasses. Let us remark also that this cluster II contains actually the majority of the documents in each subclass.

Due to this unpleasant situation, it is clear that we have to seek for other

ways of equating the clusters and the subclasses. A first idea that came in our minds was to assign to any cluster the subclass which had the largest number of documents contained in it. The subclass $US338/002$ in Table $4(a)$, for instance, is assigned to 5 clusters out of 6, therefore this first association is not appropriate to our purposes. If we assign to each subclass the cluster where it has more documents than in all the others, the Table $5(a)$ acts as a counter-example, as 4 subclasses are associated to the first cluster. Hence the simplest ideas of finding an equivalence between the clusters and the subclasses proved to lead to inconsistent results.

Thinking of more complex methods of associating clusters to subclasses we tried to improve the ones already mentioned, but instead of considering the largest number of documents, the used criterion regarded percentages, as explained in the following. Unlike the previous two tries, these methods require the construction of some new tables. For the first method the tables contain as entry at the position (i, j) the percentage of documents from the subclass j classified in the i-th cluster. To each cluster i we assign the subclass j for which the percentage at (i, j) is the largest from the row where it lies. Even if the results improve, there are still some tests, like the one represented in Table $5(a)$, where no equivalence is determined. The tables for the last proposed method contain as entry at (i, j) the percentage of documents from the i-th cluster preclassified in the subclass j. We assign to the subclass j the i-th cluster when the percentage is the largest on the column. Also in this case the results are not much better than before. The test summarized in Table $2(a)$ can be taken as a counter-example.

More complicated methods to determine equivalences between clusters and subclasses may be found, but their heuristicity is too high to risk adopting a clear verdict.

Therefore we conclude that the results delivered by our program may be interpreted as an alternative classification, whose viability is stressed by the following observation.

Consider a cluster and calculate the sum of the distances between its center and each document contained in it. Dividing this sum by the number of documents in the cluster, we obtain a so-called mean radius of the cluster. Then considering the subclasses we calculate their mean radia, too. As center of a subclass seen as a cluster we consider the mean center of all the contained documents. For each test we have noticed that the sum of the mean radia of the clusters obtained by the program is smaller than the one of the subclasses, as may be seen in Table 6. For instance, for the test represented in Table 5(a) the mean value of the mean radia of the subclasses seen as clusters is 137.92, while the same calculation concerning the clusters obtained using the algorithm give as result 117.41, which is considerably smaller, i.e. the classification we obtain concentrates data better. Some other significant results that sustain our affirmation are available in the table bellow. Only once, as may be seen in the last row of the table we have obtained almost identical results, but for the same initial data, varying a bit the parameters of the program, we have obtained a more compact classification, (which is available in the column before the last), whose mean values of the mean radia is indeed smaller than the one obtained for the given subclasses. Therefore we can say that the clusters obtained with the program are more compact than the existing ones (i.e. the subclasses) and we believe that it must deserve much more attention from the ones interested in patent classification.

tests	Table 5(a)	Table 4(b)	Table 3(b)	Table 2'(a)	Table 2(a)
mean value for the subclasses	137.92	126.58	1146.59	470.671	470.671
mean value for the clusters	117.41	83.0961	599.386	456.009	469.303

Table 6

5 Conclusions

Encouraged by the good results reported in the literature ([1]) we applied the Rose-Gurewitz-Fox algorithm in Patent Classification. We have written a program based on this algorithm in C++ which ran on 1.60 GHz Intel(R) Pentium 4 CPU machine with 1 GB RAM. In order to test our program's functionality we have constructed some artificial data sets and the obtained results proved to be good. The next step was to create data sets using the US Patent Classification System. But the results that arose after running the program on these latter data were not satisfactory in the sense that one could not discover an one-to-one mapping between the found clusters and the existing subclasses of the US Patent Classification System.

We believe that one of the reasons for this disagreement comes from the way patents were classified. The US Patent Classification System has been built manually during several decades not avoiding a touch of subjectivity. The existing classification has been made according to some criteria which cannot be entirely taking into consideration by information retrieval methods. Working with words does not always assure a complete coverage of the concepts behind them.

Though, the results delivered by the program after being run on the data sets obtained from the US Patent Classification System may be interpreted as new classification of the patents, alternative to the existing one. It is worth mentioning that this new alternative classification is more compact than the other. As an open challenge remains to understand what this new classification means for the patent databases in the real world.

Acknowledgments. The first author's work has been supported by the Federal Ministry for Economy and Technology (Germany) through the PRO INNO program. We would like to thank Dr. Ulf Bauerschäfer from the firm IP Century AG (www.ipcentury.com) for providing us the patent documents data and for fruitful discussions, comments and cooperation in the project.

We are grateful to the anonymous reviewers for their valuable remarks and suggestions that improved the quality of the paper.

References

[1] Bakker, B., Hesks, T. (1999): *Model clustering by deterministic annealing*. Proceedings of the European Symposium on Artificial Neural Networks, Bruges, April 1999, pp. 87–92.

[2] Berry, M.W., Browne, M. (1999): *Understanding search engines*. SIAM, Philapdelphia.

[3] Duda, R.O., Hart, P.E. (eds.) (1973): *Pattern classification and scene analysis*. Wiley, New York.

[4] Jain, A.K., Dubes, R.C. (eds.) (1988): *Algorithms for clustering data*. Pretience Hall, Englewood Cliffs.

[5] Gan, K.K., Kagan, H.P., Kass, R.D. (2001): *Simple demonstration of the central limit theorem using mass measurements*. American Journal of Physics 69(9), pp. 1014–1019.

[6] Gelin-Huet, C., Rose, K., Rao, A. (1999): *The deterministic annealing approach for discriminative continuous HMM design*. Proceedings of the Sixth European Conference on Speech Communication and Technology, Budapest, pp. 2717–2720.

[7] Gray, R.M., Karnin, E.D. (1982): *Multiple local minima in vector quantizers*. IEEE Transactions on Information Theory, IT-28, pp. 256–261.

[8] Gu, L., Nayak, J., Rose, K. (2000): *Discriminative training of tied-mixture HMM by deterministic annealing*. Proceedings of the Sixth International Conference on Spoken Language Processing, Beijing, pp. 183–186.

[9] Huang, Z. (1997): *A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining*. Proceedings of the SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, Tucson, Arizona, May 1997, pp. 146–151.

[10] Rissanen, J. (ed.) (1989): *Stochastic complexity in statistical inquiry*. World Scientific, Singapore.

[11] Rose, K., (1998): *Deterministic annealing for clustering, compression, classification regression and related optimization problems*. Proceedings of the IEEE 86(11), pp. 2210–2239.

[12] Rose, K., Gurewitz, E., Fox, C.G. (1990): *A deterministic annealing approach to clustering*. Physical Review Letters 65(8), pp. 945–948.

[13] Rose, K., Gurewitz, E., Fox, C.G. (1993): *Constrained clustering as an optimization method*. IEEE Transactions on Pattern Analysis and Machine Intelligence 15, pp. 785–794.

[14] Rose, K., Gurewitz, E., Fox, C.G. (1990): *Statistical mechanics and phase transitions in clustering*. Pattern Recognition Letters 11(9), pp. 589–594.

[15] Rose, K., Gurewitz, E., Fox, C.G. (1992): *Vector quantization by deterministic annealing*. IEEE Transactions on Information Theory 38, pp. 1249–1257.

[16] Rose, K., Miller, D. (1992): *Constrained clustering for data assignment problems with examples of module placement*. Proceedins of the IEEE International Symposium on Circuits and Systems, pp. 1937–1940.

[17] Shafer, J., Agrawal, R., Methe, M. (1996): *SPRINT: A Scalable Parallel Classifier for Data Minig*. Proceedings of the 22-nd VLDB Conference, Bombay, Morgan Kaufmann Publishers Inc., pp. 544–555.

[18] Bai, Z., Demmel, J., Dongarra, J., Ruhe, A., van der Vorst, H. (eds.) (2000): *Templates for the Solution of Eigenvalue Problems: A Practical Guide*. SIAM Publishers, Philadelphia.

[19] ftp://ftp.cs.cornell.edu/pub/smart/english.stop

[20] http://www.tartarus.org/∼martin/PorterStemmer