HAL
open science

An end-to-end administrative document analysis system

Hatem Hamza, Yolande Belaïd, Abdel Belaïd, Bidyut B. Chaudhuri

To cite this version:

Hatem Hamza, Yolande Belaïd, Abdel Belaïd, Bidyut B. Chaudhuri. An end-to-end administrative document analysis system. The Eighth IAPR International Workshop on Document Analysis Systems - DAS 2008, Sep 2008, Nara, Japan. pp.175-182. inria-00346019

HAL Id: inria-00346019

https://inria.hal.science/inria-00346019v1

Submitted on 12 Dec 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

An end-to-end administrative document analysis system

Hatem Hamza *,**, Yolande Belaïd**, Abdel Belaïd**, Bidyut B. Chaudhuri ***
*ITESOFT. France
**LORIA, University Nancy 2. France
***Indian Statistical Institue, Kolkata-108. India

Abstract

This paper presents an end-to-end administrative document analysis system. This system uses case-based reasoning in order to process documents from known and unknown classes. For each document, the system retrieves the nearest processing experience in order to analyze and interpret the current document. When a complete analysis is done, this document needs to be added to the document database. This requires an incremental learning process in order to take into account every new information, without losing the previous learnt ones. For this purpose, we proposed an improved version of an already existing neural network called Incremental Growing Neural Gas. Applied on documents learning and classification, this neural network reaches a recognition rate of 97.63%.

1. Introduction

Automatic processing of administrative documents has been a topic of major interest in the last few years. The huge number of such documents generated everyday makes it almost impossible for the companies to process them manually. Some automatic solutions have already been invented, starting from OCR, bank check reader, postal address reader and signature verifier, but a lot of work still needs to be done for other types of documents such as tabular forms, invoices, bills and receipts. The present paper deals with automatic processing of invoices generated in an industrial environment.

An invoice is an administrative document containing information about a commercial transaction. It contains details about a seller, a buyer, general payment information, payment details, taxes... It is a complex document which usually contains a table, one or more addresses, and other variable zones. These are the most important information to be extracted and interpreted. In order to extract this information, several steps have to be performed: scanning, OCR reading, logical and physical structure extraction, and finally interpretation. We will not focus on scanning and OCR as they are the very first part of any system, but we shall show in this paper how physical structure is extracted and then interpreted in our system.

With the increasing industrial interest in this area, many constraints such as fast processing and efficient document storage have arisen. In the last few years, several systems have been proposed for invoice analysis. Two main research directions can be identified in this area. The first one concerns data-based systems. Such systems process each document on its own, without grouping of a document to a certain class. They extract different information from documents, and analyze and interpret it based on these information. The main works that belong to this research direction are those which process tables. Table extraction and analysis is still a hard problem to solve. In [2], Belaïd presented a morphological approach for tagging tables in invoices. This approach produced good results (table cells were given the good tags in 91 % of cases). However, tables were already extracted before the tagging step. In [12], Klein presented another approach of table spotting by searching the table headers. A very good survey on table extraction can be found in [6].

The second research direction in administrative document analysis concerns model-based systems. These systems differ from the previous ones in many aspects. They not only use the information extracted from each document, but also base their analysis on a document model that is similar to the model of the current document. The challenge existing in model-based systems is the modelling itself. It can be automatic or manual. The best solution is naturally to have an automatic model, with no human intervention, but such system is rare in the literature. The most known manual modelling approaches are those which use an unfilled document (especially forms) and try to represent it with its lines, tables and cells [1] [16]. Nevertheless, while manual modelling is time consuming, automatic modelling can be very fast. It generally takes into account either the physical information that are extracted directly from the document image (lines, blocs) as done in [5], or the logical informa-

978-0-7695-3337-7/08 $25.00 © 2008 IEEE
DOI 10.1109/DAS.2008.43

IEEE computer society

Authorized licensed use limited to: UR Lorraine. Downloaded on December 10, 2008 at 11:55 from IEEE Xplore. Restrictions apply.

tion related to keywords, as done in [4] and [11].

Data-base systems are usually used in heterogeneous document flows. On the contrary model-based systems are used in homogeneous document flows, where similar documents arrive generally one after the other. In this paper, a system that can process documents from both flows is proposed. It uses data extracted from documents, document models and finally structural information.

In order to avoid human intervention at each step, this system needs to learn each new information at the document level, as well as on the structural level. For this purpose, we propose to use incremental learning. Hence, incremental neural networks are chosen as they can adapt their topology while being fed with data. We started with a neural network called Incremental Growing Neural Gas and improved it so that it becomes more related to local configurations of neurons. This neural network helped the system organizing its databases. In this paper, we will particularly focus on incremental learning.

This paper is organized as follows: the second section describes briefly the proposed document analysis system, the third section deals with docment comparison using graph probing, the fourth section introduces incremental learning and its application to administrative document processing classification and learning. The fifth section shows our experimental results. In the conclusion, we present some perspectives of this work.

2. CBRDIA: Case-Based Reasoning for Document Invoice Analysis

CBRDIA is a case-based reasoning system that can process invoices from both known and unknwon classes. Its main contributions are: (a) its ability to extract automatically the physical and logical structure of a document, (b) its ability to analyze and interpret invoices even if they are completely new to the system, (c) its ability to learn new data in an incremental way.

The first step in CBRDIA is problem elaboration which corresponds to the extraction of the physical and logical structure from the documents. In this field of document structure extraction, several extraction methods and representations have been proposed in the literature. In [4], documents are represented with attributed graphs, where nodes represent instruction and information fields (logos, lines, keywords) and the edges represent the geometrical relationships between the objects. In [3], the document model is just the set of: horizontal and vertical lines, graphic objects (logos) and text boxes (numerical, alphabetical or alphanumerical text). In [17], business letters are represented with attributed directed graphs. Each node corresponds to a logical object (sender, recipient, date...). In [13], Doermann used an attributed graph to describe the logical structure of documents. Nodes represented the logical information (zones, tables, titles, blocks) whereas the edges represented the relative positions between these information.

Some works were also concerned with automatic model extraction [9] whereas others used a manual model that is registered by a user. It is obvious that automatic modelling is much more desirable (as it is quicker) in an industrial environment. Manual modelling is still used in industry, but it should normally be used a complement to automatic modelling.

We wanted to have an automatic modelling phase in CBRDIA. We also had to avoid having a document model that is very specific to a certain document, this is the reason why we did not use any absolute coordinate in our final models. The documents that are used in our approach are real world documents taken from a document processing chain. First, they are OCRed thanks to some commercial softwares combined with other local built OCRs. The OCR output is then all the document words with their coordinates in the document. These words have to be organized in a more logical way. For this, we create groups of information given below:

1. Words: which are returned by the OCR. They are given the following attributes: position (top, left, right, bottom), tags (alphabetical, numerical, alphanumerical).

2. Fields: which are groups of neighbor words aligned horizontally. They are given the following attributes: positions and tags. The field tags correspond to the concatenation of the tags of the words composing the field.

3. Horizontal lines: which are groups of neighboring fields. They are given the following attributes: position, pattern. A pattern is the concatenation of the tags of the fields composing the horizontal lines.

4. Vertical blocks: which are groups of fields aligned vertically. They are given the following attributes: position coordinates.

From this physical re-structuring of the document, we extract now the logical structure. We have, noticed through our observations on thousands of documents that two kinds of structures exist in invoices. The first one is the keywords structures (KWS) that are based on the extraction of some keywords. This extraction uses dictionaries related to the field of invoice processing (example: words like Amount, VAT, Total...). The second type of structures is tables, which are very important structures in administrative documents. As tables correspond generally to a repetition of a pattern, these structures are called pattern structures (PS). Once these two types of structures are extracted, the document model takes them into account as well as their relative

Authorized licensed use limited to: UR Lorraine. Downloaded on December 10, 2008 at 11:55 from IEEE Xplore. Restrictions apply.

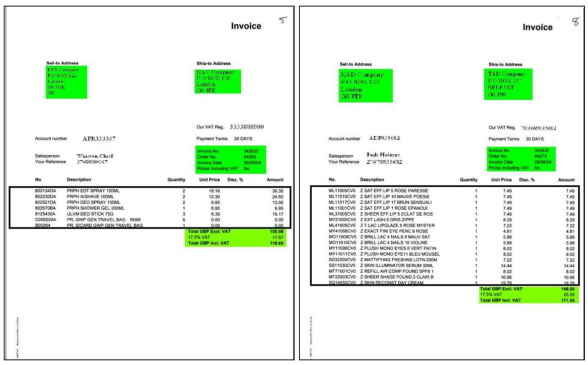

Figure 1. Two documents from the same class

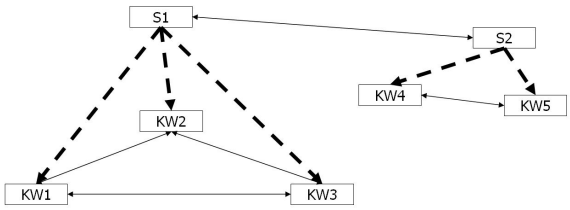

Figure 2. A document model

positions (top, left, bottom, right). Further details about our document model extraction, and especially on table extraction can be found in [9].

The final document model is a graph of the different structures of the documents. This representation allows us not only to describe a document, but also a whole class of documents. In this way, whenever a document from the class 'X' is presented to the system, it can be directly recognized as belonging to this class. The example in the figure 1 shows clearly how two document can be represented with the same model.

Figure 2 shows a document model. This document is composed of two KWS. It can be clearly seen that the nodes are the document keywords, or the labels of the structures, whereas the edges describe the relative positions between the elements.

Once the document model is extracted, global problem solving starts. It consists of checking if a similar document is available in the document database (using graph probing or edit distance). If it is the case, then the solution of the nearest document is applied on the current document. Otherwise, local solving is used. It consists of finding a solution for every structure (KWS or PS) in the document independant of others. Similarly, as in global solving, the system looks for similar structures in the structure database and tries to apply the solution of the nearest ones on the current structures. Figure 3 shows the flow of this approach. Further details can be found in [9].

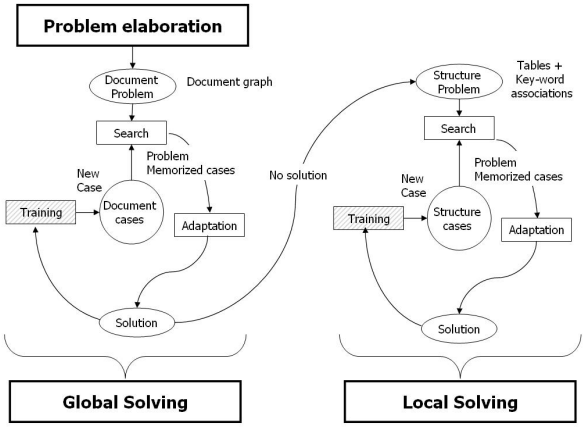

Figure 3. CBRDIA flow

3. Graph matching using graph probing

In CBRDIA, every document is to be matched with the documents of the database. Since our documents are represented by graphs, graph matching techniques have to be used. Many graph similarity measures exist. Edit distance as well as the maximum common subgraph distance can be employed, but time and complexity are factors that lead us to think about a faster similarity measure.

Graph probing distance is a graph dissimilarity measure that was first presented in [14]. It is a fast and fairly accurate technique of graph comparison. It has also a direct relation with graph edit distance. Let G1 and G2 be two graphs, then $d_{graphprobing}(G1, G2) \leq 4.d_{editdistance}(G1, G2)$. Quite often graph probing is a good approximation of graph edit distance. Its principal drawback is the fact that if $d_{graphprobing}(G1, G2) = 0$, then G1 and G2 are not necessarily isomorphic.

Graph probing is based on the computation of the frequency of each vertex and each edge in its graph. Let A, B, C be the nodes of G1, and B, C, D be the nodes of G2. First, we compute the frequency of A, B, C in G1, and do the same with the nodes of G2. The probe on nodes is then: $Pb1 = \sum |freq(N_{G1}) - freq(N_{G2})|$, where $freq(N_{G1})$ and $freq(N_{G2})$ are respectively the frequencies of a node N in G1 and G2.

On the other hand, we have to calculate the probe on edges. For every node, we extract its edge structure. If a node N has an edge with the tag (top, left) and another one with the tag (top, right), then the edge structure of N is (top:2, left:1, bottom:0, right:1). This is done for every edge in the graph. The probe over edges is then

Authorized licensed use limited to: UR Lorraine. Downloaded on December 10, 2008 at 11:55 from IEEE Xplore. Restrictions apply.

$Pb2 = \sum |freq(E_{G1}) - freq(E_{G2})|$, where $freq(E_{G1})$ and $freq(E_{G2})$ are respectively the frequencies of an edge structure E in G1 and G2.

The total probe is then: $Pb = Pb1 + Pb2$.

Graph probing is applied in this way to every new document in order to find the most similar document in the database.

4. Incremental learning in CBRDIA

Now that new cases of documents have to be learnt, incremental learning is adopted. It allows the system to use the solved cases in the future and to avoid the same processing for every similar document. Incremental learning is used as the folowing: every new case has to be learnt and retained by the case base. Moreover, similar cases have to be grouped together in order to make the comparison between an incoming case and the case base more accurate.

To find an approach of incremental learning, we focused on incremental neural networks. The earliest incremental neural networks were the Growing Cell Structures (GCS [7]), followed by the Growing neural Gas (GNG). Then, many other variations were built on these two networks. The Hierarchical GNG (TreeGNG) is a network that builds classes over the classes given by the GNG. Similarly, the Hierarchical GCS (TreeGCS [10]) uses the same principle. Other types of incremental neural networks are those which use self organizing maps. One has to make the difference between incremental neural networks which perform incremental learning (GNG, IGNG) and incremental neural networks which are just incremental because they can add or remove neurons. We will just introduce IGNG in this paper as it is the method we are using.

4.1. Incremental Growing Neural Gas

The IGNG is an improvement of the GNG in some aspects (algorithm 1). As shown in [15], IGNG gives better results for online and in incremental learning. This can be explained by the fact that IGNG creates neurons only when a new data is very far from the already created neurons, contrary to the GNG which creates neurons periodically. IGNG has also better memory properties. This means that when a new class of data having different properties appears, the IGNG can really adapt its topology without loss of the previous information, whereas the GNG can lose some of its already created neurons.

This neural network IGNG suffers however from the choice of the threshold S. In their original paper, Prudent et al. proposed to initialize S at the standard deviation of the whole database for which classification is done. This is in our opinion contrary to the principles of incremental learning as we do not know $apriori$ which kind data is coming later.

```
S, ε_b, ε_n, a_edge, a_neuron_max : IGNG parameters;
While (a stopping criteria is not found) do
    Take an input signal E and find its nearest neuron n_1.
    If ( the network is empty or d(E, n_1) > S) then
        create a new embryon neuron ω_new = E
    else
        Find the second nearest neuron n_2
        If ( There is only one neuron in the network or
        d(E, n_2) > S) then
            create a new embryon neuron ω_new = E
            create a new edge between n_1 and E.
        else
            increment the age of all edges coming
            from n_1.
            n_1 += ε_b · d(E, n_1)
            n_m += ε_n · d(E, n_m), (m are the neigh-
            bours of n_1 )
            If ( n_1 et n_2 are connected ) then
                age_{n_1→n_2} = 0
            else
                create an edge between n_1 and n_2
            end If
            Increment the age of all the neighbour neu-
            rons of n_1
            For each embryon neuron do
                If ( age(neuron) > a_neuron_max)
                then
                    embryon neuron becomes ma-
                    ture
                end If
            end For

        end If
    end If
    For each edge do
        If ( age(edge) > a_edge) then
            remove edge
        end If
    end For

done
```

Algorithm 1: IGNG

4.2. IGNG improvement

A study on the threshold S

The first point on which we worked was to try to be free from the choice of the threshold S. The first constraint is that the only information available at a time T is the information about the already processed data. Moreover, we cannot use the whole previous data to determine the class of the new

Authorized licensed use limited to: UR Lorraine. Downloaded on December 10, 2008 at 11:55 from IEEE Xplore. Restrictions apply.

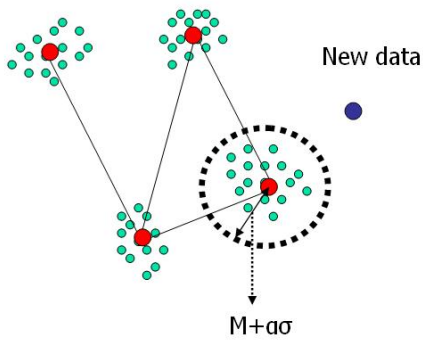

Figure 4. A new data creates an embryon neuron

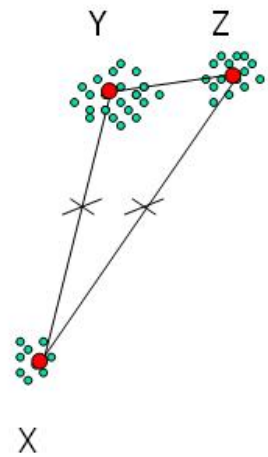

Figure 5. A neuron that should not be deleted, even if it is not connected to any other neuron

data. The solution is to use some local information related to each neuron. Let:

- N be the number of IGNG neurons at T.
- E be an entry.
- m_i be the average distance between every element in a class i and its representative neuron n_i. σ_i the standard deviation of these distances.

It is logical to say that E belongs to a class i if $d(E, n_i) < m_i$. In order to be more flexible, we propose that the threshold S becomes: $S = m_i + \alpha.\sigma_i$, where n_i is the nearest neuron to E. By taking into account the mean and standard deviation of this class, we are using intrinsic parameters related to this class, not to the whole data. Two cases typically occur:

- the new data is close enough to the nearest class (meaning $d(E, n_i) < m_i + \alpha.\sigma_i$), this data will belong to the class i and the neuron n_i is updated.
- the new data is too far from its nearest class. In this case, a new neuron is created (embryon neuron), and becomes effective in classification only if its age exceeds $a_{neuron_{max}}$.

Figure 4 shows a typical case where the nearest class is too far from the new data. This new data will then create an embryon neuron.

Node deletion

The second point of interest is the condition of node deletion. In GNG and IGNG, an edge is removed whenever its age exceeds a threshold a_{edge}. Then, if a neuron is not connected to any other neuron, it is removed as well. However, depending on the application, one may not want to remove neurons that are not connected to others, as these single neurons may represent an important information (eventhough it can be very rare). For example, in our application, some rare invoices can be processed from time to time, and can in this way form neurons that are very far from the other neurons. A simple example is shown on figure 5. The neuron representing the rare data (X) is connected to two other neurons (Y and Z) which have many more data compared to X. Every time a data is attributed to Y or Z, the edges (X-Y) and (X-Z) are incremented. When $age(X - Y) > a_{edge}$ and $age(X - Z) > a_{edge}$, these edges are removed. In the classical scheme, X will also be removed. Moreover, its data will be assigned to its nearest neurons in the network (Y and Z). Then, even if Y and Z are not representative of the data associated with X, their data will include new data very different from their original data.

We propose in this case to examine the distance between X and its nearest neuron, let this distance be d(X,Y). If $d(X, Y) > \beta \cdot ((m_y + \alpha.\sigma_y) + (m_x + \alpha.\sigma_x))$, then this X has to be kept in the network, even if it has no connection with the other neurons. Otherwise, this neuron can be removed and its data can be assigned to the other neurons. The β factor can be chosen by the user, depending on the application.

From now on, the Improved Incremental Growing Neural Gas will be noted I2GNG.

4.3. Adaptation to the case of graph classification

In [8], Gunter and al. proposed a method of adapting Self Organizing Maps (SOM) to graph classification. This method was used to digit classification and was based on the computation of the edit distance between graphs. Ev-

Authorized licensed use limited to: UR Lorraine. Downloaded on December 10, 2008 at 11:55 from IEEE Xplore. Restrictions apply.

ery formula in the SOM algorithm was then adapted based on the edit path between any two graphs. Here is a simple explanation of the idea:

let G1 and G2 be two graphs. Let d(G1,G2) be the edit distance between G1 and G2. This distance corresponds to the cost of some additions, deletions or substitutions of nodes or/and edges which transform G1 into G2.

$$d(G1, G2) = \sum cost(editing)$$

In the vector domain, when the distance between a vector X and vector Y is d(X,Y), it is easy to transform X by $\epsilon(X, Y)$, ϵ being a real quantity. The same operation in the graph domain means that G1 has to be modified by $\beta = \epsilon \cdot d(G1, G2)$ (equivalent to $Neuron = \epsilon_b.(Neuron_{nearest} - entry)$). Modifying G1 by β means that we have to apply only β edit operations on G1. As we already know the edit path that allowed us to compute the distance between G1 and G2, β corresponds just to a part of this edit path. Modifying G1 becomes in this way an easy task as we just have to find the edit operations which cost approaches β as much as possible. More elaborate details can be found in [8].

Adapting the I2GNG formula using the principles cited above allows us now to classify graphs or trees using I2GNG.

5. Experiments

The first part of these experiments shows the results of the whole CBRDIA system. The other experiments are only related to incremental learning using different datasets.

5.1. Experiments on CBRDIA

CBRDIA was tested on 950 invoice documents taken from invoice processing chain of ITESOFT. They are divided in 2 groups:

- the first one contains 150 documents where each one has a similar case in the document database: this is used to test global solving. The document database contains 10 different cases;
- the second one contains 800 documents for which no associated case exists in the document database. Hence, local solving will be applied on these documents.

The results are described thanks to three different measures as in 1. In this equation, X can be a document, a KWS or a PS.

$$R_X = \frac{|found\ solutions\ |}{|solutions\ in\ ground\ truth\ X|}. \quad (1)$$

Global solving produced 85.29% of good results whereas local solving produced 76.33%. More detailed results can be found in [9].

5.2. Experiments on synthetic data

These experiments were carried out on synthetic 2D data in order to test the improvements of I2GNG.The first experiment consisted of testing the incremental behaviour of the improved I2GNG. Two distribution were submitted to the I2GNG: the first one is circular, and the second one is triangular. We wanted to see if the second distribution could alter the I2GNG learnt on the circular one. The figure 6 shows the obtained results and confirms that the improved I2GNG, as the original one, has good memory properties (which is very important in our application).

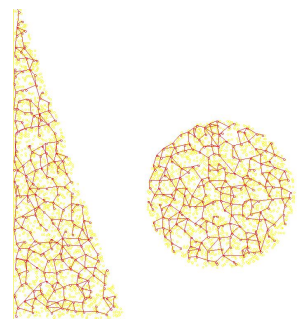

Figure 6. The improved I2GNG does not lose its already learnt neurons

The second experiment consisted in checking whether the order of arrival of the data can have an influence on the convergence of the I2GNG. Similarly, we took the previous distributions, but data was given to the I2GNG randomly from anyone of them. The results in figure 7 show that the I2GNG is robust against the random arrival of data (which is a realistic case in our application: every invoice can have no relationship with the previous and following ones).

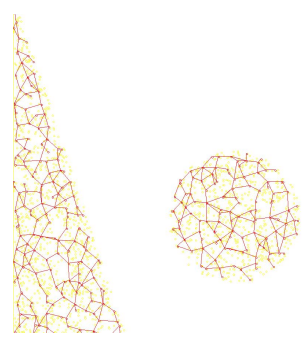

Figure 7. The improved I2GNG is robust to the order of arrival of data

Authorized licensed use limited to: UR Lorraine. Downloaded on December 10, 2008 at 11:55 from IEEE Xplore. Restrictions apply.

5.3. Experiments on the MNIST database

Another experiment was performed on the MNIST database. In this experiment, the learning examples were given to the I2GNG progressively, and tests were performed after each 10000 images. The distance used to compute the similarity between each pair of images is the Euclidian distance. Even if this is not the best distance to use in the case of image comparison, we used it in order to evaluate the I2GNG capacity of classification. Each learning sample was given only once to the I2GNG. The results are shown in table 1.

samples	recognition
10000	88.45%
20000	91.02%
30000	92.58%
40000	93.66%
50000	94.06%
60000	94.29%

Table 1. I2GNG results on the MNIST database.

The obtained results are far from the best results obtained on the MNIST. However, one should notice that these results were obtained after one single pass of the data. The closest work to ours on the MNIST is a work done by Wilder[1] using K nearest neighbours classifier, after preprocessing the database images. The error rate of this work reached 1.22%.

With similar pre-processing steps, we can expect identical results with the I2GNG.

5.4. Experiments on administrative documents

Our experiments were made on a dataset of real documents (invoices) taken from a real invoice processing chain. Every invoice was modeled with its graph and then submitted to the I2GNG. The complexity of these documents is variable. Whereas some documents are very clean and have almost no OCR errors, others can be degraded and have very few key-words to be identified.

The dataset was divided in two parts: a learning set (324 documents) and a testing set (169 documents). 8 classes of invoices were used for this purpose. We chose this strategy of I2GNG evaluation as the learning procedure helps in knowing about the incremental capabilities of the modified I2GNG applied to graphs, whereas the testing phase helps knowing about its classification properties.

[1] http://oldmill.uchicago.edu/ wilder/Mnist/

We performed two different series of tests. The first one is concerned with the influence of α, the second one concerns the influence of the threshold age of neurons (above which neurons become mature). Table 2 gives these results.

a_{edge}	neurons	rec	α	neurons	rec
10	14	99.40%	0.5	10	98.22%
20	18	97.63%	1	15	98.22%
30	18	97.63%	1.5	12	98.81%
40	16	98.22%	2	14	98.81%
50	16	98.22%	2.5	12	99.40%
60	16	98.22%	3	18	97.63%

Table 2. influence of α and a_{edge}

The results show that the number of neurons is always greater than the number of classes (8). This is due to the variations found in these classes. Representing one class with several neurons is not a problem as far as a neuron is not shared by two classes. In the training process, we tag manually the obtained neurons (by giving them the name of the class they represent).

The obtained results are very encouraging. They mean that the I2GNG with our improvements is working well. As shown in table 2, the bigger α is, the more neurons we obtain. This can be explained as the following:

- when α is big, the threshold $(m+\alpha.\sigma)$ for each neuron is also big. A new class is created if and only if it is outside the range of an existing neuron. Neurons are then not close to each others. Their ages increase quickly and they become mature quickly too.
- on the other hand, when α is small, neurons are very close to each other. One class can be represented by high number of neurons, few of which become mature because of the high competition among classes.

6 Conclusion

In this paper, we presented the different steps composing the system CBRDIA. This system and its incremental learning part have been implemented and tested on real data.

An existing neural network was improved and extended to graph classification. The obtained results are satisfying, but some work still needs to be done in order to improve the performances of the I2GNG. Two studies are being done. The first one concerns the automation of the choice of the maximum ages of edges and neurons. These parameters have an influence on the final results. The second study concerns α the choice of which can be done using some characteristics of the studied neuron. For example, we can use the density of a class, its entropy or other descriptors to get an adaptive α.

Authorized licensed use limited to: UR Lorraine. Downloaded on December 10, 2008 at 11:55 from IEEE Xplore. Restrictions apply.

References

[1] H. Arai and K. Odaka. Form processing based on background region analysis. In *ICDAR*, pages 164–169, 1997.

[2] Y. Belaïd and A. Belaïd. Morphological tagging approach in document analysis of invoices. In *ICPR*, pages 469–472, 2004.

[3] F. Cesarini, E. Francesconi, M. Gori, and G. Soda. Analysis and understanding of multi-class invoices. *IJDAR*, 6(2):102–114, 2003.

[4] F. Cesarini, M. Gori, S. Marinai, and G. Soda. Informys: A flexible invoice-like form-reader system. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(7):730–745, 1998.

[5] P. Duygulu and V. Atalay. A hierarchical representation of form documents for identification and retrieval. *IJDAR*, 5(1):17–27, 2002.

[6] A. C. e Silva, A. M. Jorge, and L. Torgo. Design of an end-to-end method to extract information from tables. *IJDAR*, 8(2-3):144–171, 2006.

[7] B. Fritzke. Growing cell structures–a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460, 1994.

[8] S. Günter and H. Bunke. Self-organizing map for clustering in the graph domain. *Pattern Recognition Letters*, 23(4):405–417, 2002.

[9] H. Hamza, Y. Belaïd, and A. Belaïd. Case-based reasoning for invoice analysis and recognition. In *ICCBR*, pages 404–418, 2007.

[10] V. J. Hodge and J. Austin. Hierarchical growing cell structures: Treegcs. *Knowledge and Data Engineering*, 13(2):207–218, 2001.

[11] Y. Ishitani. Model based information extraction and its application to document images. In *DLIA*, 2001.

[12] B. Klein, S. Gökkus, T. Kieninger, and A. Dengel. Three approaches to "industrial" table spotting. In *ICDAR*, pages 513–517, 2001.

[13] J. Liang and D. S. Doermann. Logical labeling of document images using layout graph matching with adaptive learning. In *Document Analysis Systems*, pages 224–235, 2002.

[14] D. P. Lopresti and G. T. Wilfong. A fast technique for comparing graph representations with applications to performance evaluation. *IJDAR*, 6(4):219–229, 2003.

[15] Y. Prudent and A. Ennaji. A new learning algorithm for incremental self-organizing maps. In *ESANN*, pages 7–12, 2005.

[16] Y. Y. Tang and J. Liu. Information acquisition and storage of forms in document processing. In *ICDAR*, pages 170–174, 1997.

[17] H. Walischewski. Automatic knowledge acquisition for spatial document interpretation. In *ICDAR*, pages 243–, 1997.

Authorized licensed use limited to: UR Lorraine. Downloaded on December 10, 2008 at 11:55 from IEEE Xplore. Restrictions apply.