Administrative Document Analysis and Structure

Abdel Belaïd, Vincent Poulain D'Andecy, Hatem Hamza, and Yolande Belaïd

Abstract. This chapter reports our knowledge about the analysis and recognition of scanned administrative documents. Regarding essentially the administrative paper flow with new and continuous arrivals, all the conventional techniques reserved to static databases modeling and recognition are doomed to failure. For this purpose, a new technique based on the experience was investigated giving very promising results. This technique is related to the case-based reasoning already used in data mining and various problems of machine learning. After the presentation of the context related to the administrative document flow and its requirements in a real time processing, we present a case based reasonning for invoice processing. The case corresponds to the co-existence of a problem and its solution. The problem in an invoice corresponds to a local structure such as the keywords of an address or the line patterns in the amounts table, while the solution is related to their content. This problem is then compared to a document case base using graph probing. For this purpose, we proposed an improvement of an already existing neural network called Incremental Growing Neural Gas.

1 Introduction

One could think that paper is something from the past, that documents will all be "electronic" and that our offices and world is becoming paperless. This will be true one day, but it will take several decades or centuries because it is not only a

Abdel Belaïd
LORIA, University of Nancy 2
e-mail: abdel.belaid@loria.fl

Vincent Poulain D'Andecy
e-mail: Vincent.PoulaindAndecy@itesoft.com

Hatem Hamza
e-mail: Hatem.Hamza@itesoft.com

Yolande Belaïd
e-mail: yolande.belaid@loria.fl

M. Biba and F. Xhafa (Eds.): Learning Structure and Schemas from Documents, SCI 375, pp. 51–71.
springerlink.com

technological mater but a real cultural issue. Even though they all surf and chat over the Internet, our kids still learn how to read with paper books and how to write with pen and paper. Whatever the future, if we just have a look today at our administration services such as public organisations and services (city hall), social security services (health insurance, family benefits, hospitals, pension funds), large companies (bank, insurance) etc., they handle each day a large volume of documents from their citizens and customers. On the other hand they face several challenges in order to reduce the charges and increase the customer satisfaction. They do look for processing times reduction; process error reduction and delivery of new services that can share quickly electronic information in an internet or a mobile context.

One solution is to turn the administrative paper flow into an electronic document flow. That means digitize the documents, extract automatically by document analysis techniques some information from the content of these digital images, push the extracted indexes into a business system like an ERP (Enterprise Resource Planning) and finally archive the indexed images into a DMS (Document Management System).

For the last 20 years, several software editors have proposed intelligent systems to automatically read the administrative documents. The current most famous companies are EMC-Captiva, ReadSoft, Top Image System, Kofax, Itesoft, A2IA, ABBYY, Nuancet, Pegasus, Mitek, Parascript,

The performance of these systems is a good enough fit for some business objectives on few documents such as cheques, invoices, orders or forms but the performance deteriorates quickly when the system faces the complexity and the large variability of administrative document types. Basically an administrative document is a textual document, usually a form. The issue is that the information to extract is very heterogeneous:

- The extracted information can be high level information like a document class (this is an invoice, a complain, a form) or low level information such as fields (invoice number, social security ID, cheque amount) or tables (list of ordered items).
- The information can be machine printed (barcodes, characters, symbols, logos), hand-printed (script, marks) in a constraint frame or handwritten (cursive) with no constraint.
- The information can be very structured like forms. For one form each field is always located at the same physical position. Notice that there is a vast amount of different administrative document structures. Then if the automation of one form can be quite simple, the automation of hundreds of forms in a single flow becomes a bottleneck.
- The information can be semi-structured like bank cheques or invoices. The document looks like a structured document but the field location is floating, respecting only a logical structure. Each supplier prints the document in their own way but respecting (almost) for instance the rule <invoice number is closed to the top of the page beside the keyword ”invoice no”>.
- The information can be unstructured like a free text in a natural language (handwritten mails).

- The information can be explicit, directly readable within the document, or implicit, i.e. not directly readable but inferred from readable information, eventually combined with customer context data.
- The information can be isolated and easy to locate (printed or written on a white background) or with overlapping and then hard to segment (printed on a textured background or written over a comb or a printed text,).
- The information can be in color while most of the industrial capture systems produce binary images.

All the proposed systems rely on 2 important kinds of technologies:

- Classifiers to convert the pixels into symbols: OCR (Optical Character Recognition) for machine printed symbols, ICR (Intelligent Character Recognition) for hand-printed characters [1], IWR (Intelligent Word Recognition) for handwritten cursive words [3] [2], Pattern Matching [4] for some graphic elements, Logo Recognition [5] [6], etc.
- Segmentation and extraction strategies to locate and interpret the field to extract. We identify two different techniques: the template matching approach [7] [8] and full-text approach [26] [31].

In the template matching approach, the extraction is driven by a mask. The mask defines an accurate physical location (bounding box) for each field to extract within the image. The system interprets the document in applying the classifier (OCR, ICR) to the snippet framed by the bounding box. In the industrial systems, the mask description is usually done by manual settings on a graphical interface. This is easy to handle, very efficient for fixed-structure forms, but is time-consuming when the user has to define many templates. Some researchers have proposed automatic model methods. The system builds itself the template by detecting graphical features: Duygulu [9] uses horizontal and vertical lines; Mao [10], Ting [12] use lines, logo, text lines; Hroux [13] uses pixel density; Sako [14] uses keywords ; Cesarini [15] uses both keywords and graphical features, Belaïd [16] uses connected component sequences. These features are exploited by a physical structure analysis algorithm which can perform top-down analysis as RLSA based techniques [17]; bottom-up analysis for example, X-Y cut [18] X-Y tree [9] [7]; and generally said more efficient, hybrid approaches like Pavlidis Split & Merge [19].

When the document flow contains heterogeneous documents that can not be described by a single template, then a classification step is required to select the adapted model. Some classification techniques rely on the same graphical features as described above. For instance Mao [10] uses also the lines to discriminate the document classes; Heroux [13] uses a vector of pixel density pyramid; Sako [20] describes a document class with keywords. By extension, text classification strategy with vector of words [21] can be used. For instance EMC, Itesoft and A2IA products integrate both graphical and textual classifiers. With these features many categorization algorithms have been explored like decision trees, nearest neighbour [22] [12], neural networks [23] [13], prototype-based methods, support vector machines [24].

The automatic model is an interesting method but limited to one or few similar document classes. In fact the features and the document structure analysis hardcode the application domain: invoices [15], orders [16], cheques, forms or tables [10]. Therefore if the method allows a good flexibility in the domain scope, it can not manage heterogeneous flows without an important development invest. The full-text approach answers to this issue.

In the full-text approach, the extraction is driven by the data itself and a set of rules (knowledge base) to find logical labels [25]. The rules can be parametrized by a user or trained on labeled documents like Cesarini [26] who trains an M-X-Y tree to learn the table structures. Most of the approaches are formal grammar based as in Couasnon [28] or Conway [29]. Niyogi [30] presented a system called DeLoS for document logical structure derivation. Dengel [31] described a system (DAVOS) that is capable of both learning and extracting document logical structure. Grammar and tree-representation seem to be a major area of research, the technique can even work for specialized tasks such as table form recognition [20]. A few systems try to learn the relationship between physical and logical structures [37] that do not focus especially on logical labeling. These systems can outperform classical rule base systems because they use more flexible models, which can tolerate variations of the structure [34]. In most recent works, new human brain based approaches have been developed: Rangoni [35] combines learning and knowledge integration by using a transparent neural network;

However all these systems face the same problems:

- Any segmentation and extraction strategies need an expert to describe the documents and ”teach” the system. These settings or training are still time consuming and not often final-user friendly.
- Only few solutions address the full scope of heterogeneous information.
- The solutions depend on the intrinsic performance of the character classifiers. These classifiers are very efficient in standard contexts but usually collapse in noisy context.

We proposed in [38] a case-based reasoning system to model and train the knowledge for the recognition of administrative documents corresponding to different kind of invoices. They have a typical format that changes depending on the issuing company or administration. It may vary and evolve even within the same company. To cope with these changes, we seek to develop a system able to rely on experience to recognize new samples. The case-based reasoning seems to be a methodology capable of meeting this challenge. We will first describe the methodology, then we will show how we exploit it in the case of invoices.

This chapter will be organized as follows. Section 2 provides a general definition of CBR, explaining its various components. Then, Section 3 shows the use of CBR for document analysis. The proposed approach is described in Section 4. Experiments on invoices are detailed in Section 5. Finally, we conclude and give some perspectives of this work in Section 6.

2 Case-Based Reasoning

The case-based reasoning (CBR) is a reasoning paradigm that uses past experiences to solve new problems [39]. It is applied today in all areas where we need to use or synthesize past experiences to propose new solutions. Early work in CBR has been proposed in 1983 by Kolodner [41], which created a system of questions and answers using a knowledge base already established.

CBR is based on several steps we will detail in this paper. Several models have been proposed to define the CBR (see Fig. 1), but the most common is that of Aamodt and Plaza [39]. It consists of four steps: research, reuse, revision, retention (or learning). A preliminary step, also needed, is the formulation of the problem. In this work, we combined stages for reuse and review a step conventionally called "adaptation". This is also done in most researches in CBR.

2.1 CBR Terminology

In CBR, a problem is posed by the user or the system. That is the part to resolve. A case is the set formed by the problem and its solution.

$$case = \{Problem, Solution\}$$

We call "target case" the case to solve and "source case" the case from the case base used to solve the target case. The base case is the core of the system cases. These are either given by the user, or by the system automatically enriched.

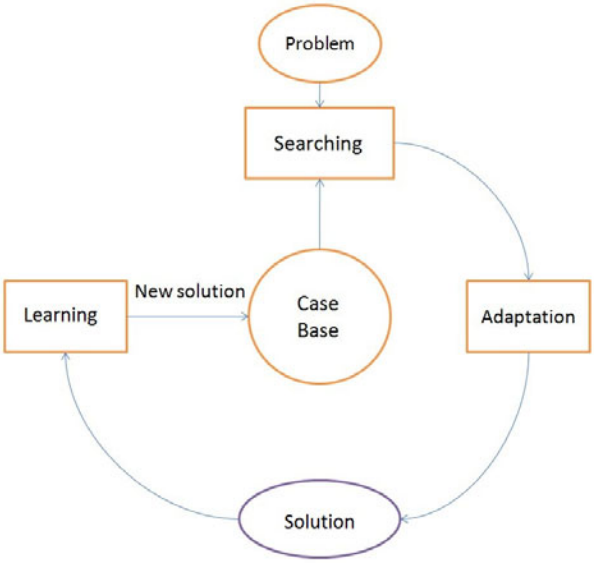

Fig. 1 CBR paradigm

2.2 Problem Elaboration

This phase involves the indices and problem descriptors extraction, and their representation. It can be done either by the user who, based on his data, constitutes the problem to solve, or by the system which knows itself how to elaborate a problem based on available information. For example, in the system FAQFinder [40], the system input is a question given by the user. The question is then processed by the system which extracts the problem (extraction of the most significant terms, removing the less informative terms). The final problem in FAQFinder is a vector representing the query.

2.3 Similar Case Search

Similar case search is one of the most important parts of CBR. It involves taking into account the representation of cases, a measure of similarity between cases and the choice of one or more close cases. The similarity measure between cases is often limited to a similarity measure between problems. It depends directly on the data representation and complexity. Once the similarity measure appropriate to the problem is chosen, it becomes easy for each new problem, to find similar cases in the case base.

2.4 Adaptation

The adaptation is to find a solution to the target problem from a solution given to the source problem . The solution to the source problem should be kept or modified to allow the proposal of a new solution to the target problem. Two major categories of adaptation exist in the literature:

- Structural adjustment: it tries to apply the solution of the source case through some changes, taking into account the differences between source and target problems, on the target problem.
- Derivational adaptation which traces how the source solution has been proposed to produce the solution to target. This is particularly true for planning applications for example.

These two broad categories can be broken down into several types of other adaptations[42]. The simplest, and it's one we'll use is the zero adjustment. It consists, from the solution of the problem source, to "paste" the solution on the target problem. This adaptation, although it is very simple, solves problems as the case of documents for example.

2.5 Learning

The role of the learning is to integrate new solved and revised cases in the case base. This process must be designed to avoid filling the database with each new

case resolved (redundancy can be dangerous for the system as far as it can slow it down considerably). It allows the system to manage its knowledge. We propose here an interesting solution for classifying the case base. We stand where the case base is constantly fed with new cases resolved as different, and we try to build a system to classify these cases so as to always have quick access to the database.

3 CBR for Document Image Analysis: CBRDIA

In an administrative Document Image Analysis (DIA) system, it often happens that documents are processed in batches. We call batch a set of similar documents, from the same supplier, the same company or administration. Documents in a batch all have the same characteristics, they are represented in the same manner and only the specific content of each document is different (for example, the cell contents in a form). Currently, in a company, a batch requires a user intervention to model the material on this batch. From the model, it becomes easier for the system to extract the desired information in the document.

The first problem is to try to model documents automatically in a batch and use this model later on the batch. The second problem is more complex. The system must also be able to handle heterogeneous documents. In this case, two configurations are possible:

- The first, and it is the easiest, is the case of the document for which you can find a similar model. This means that documents similar to this document have already been treated before, and it can speed up the processing. Documents in Figure 2 have exactly the same structures. As well the tables as the address structures and payment are similar. It is clear then that knowing the general model of this batch, it becomes very easy to treat all documents belonging to this batch.
- The second is more complex. This is the case of a completely new document, for which no existing model can be associated, and must of course be analyzed. The easiest solution is to appeal to a user who will help to extract the necessary information.

Fig. 2 Two invoices from the same batch. The model of these documents is the same

These two problems raised other issues as important:

- how to model documents, what information must be represented in this model,
- how to take advantage of existing templates,
- which approach to take for that user intervention is minimal.

In our application, the system tries to analyze the documents individually, without resorting models of existing documents. The example of invoices in Figure 3 shows two invoices from two different batches. It is clear that both invoices should not (and can in fact not) be treated in the same way. Special treatment should be performed on each of them.

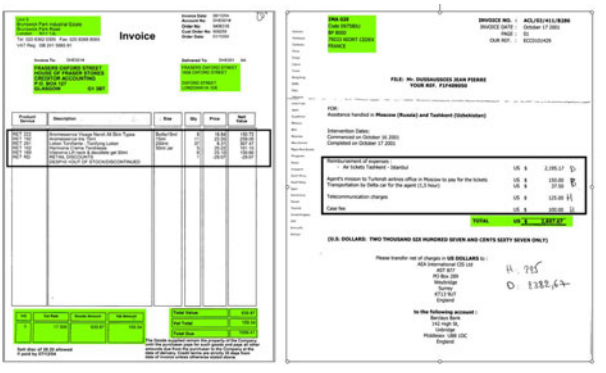

Fig. 3 Two invoices from different batches

The ideal solution would be to try to enjoy the experience of the system, instead of the user. As to treatment, the system accumulates the experience of document analysis. After a while, it is certain that the system has treated so many different documents that the knowledge it gained could be used to analyze any type of documents. We must therefore take advantage of that knowledge. This is the main idea of the solution to the problems mentioned above. We will therefore introduce a system that not only deals with documents, but also learns as you go.

For this reason we chose to use CBR for the proposed system. First, the existing mode of reasoning in CBR is well suited to the needs of a system for document analysis. Indeed, CBR can benefit from previous experiences to offer new solutions to new problems. In the case of our application, a new problem is simply a new document to be analyzed. The fact that this document be in a batch or completely new does not change the fact that we must be able to analyze and interpret. Previous experiences are therefore other than documents previously modeled, analyzed and interpreted.

4 The Proposed Approach

Figure 4 shows the block diagram of our approach. It consists of two CBR cycles. For each new document, the problem is first extracted. This is compared with document problems of the case base.

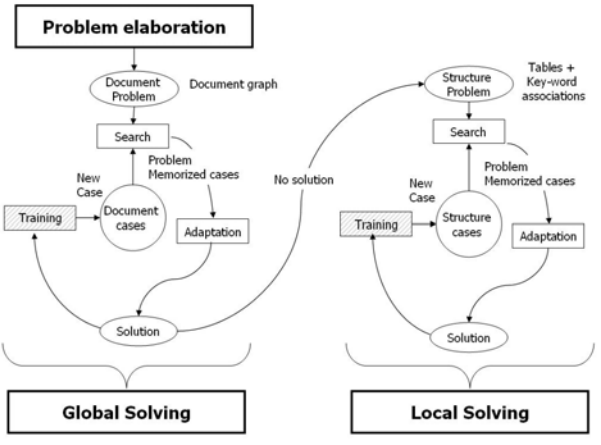

Fig. 4 CBRDIA flow

- If a similar problem exists in the database, then the solution of the nearest (source problem) is applied to the problem of the target document.
- If no similar problem exists in the case base, then we proceed to the next CBR cycle. The structure problems are used. Each structure issue is compared with the problems of the structure case base. The solutions of the closest problems are then applied to the problem of target structure. Through this process we can obtain a complete solution for the document problem. So we can inject this resolved case in the document case base.

4.1 Document Structures

The document structures constitute in our approach the problems in CBR terminology. 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 semantic dictionaries related to the domain of invoice (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 positions (top, left, bottom, right). Further details about our document model extraction, and especially on table extraction can be found in [43].

4.2 Problem Representation

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 Figure 5 shows clearly how two documents can be represented with the same model.

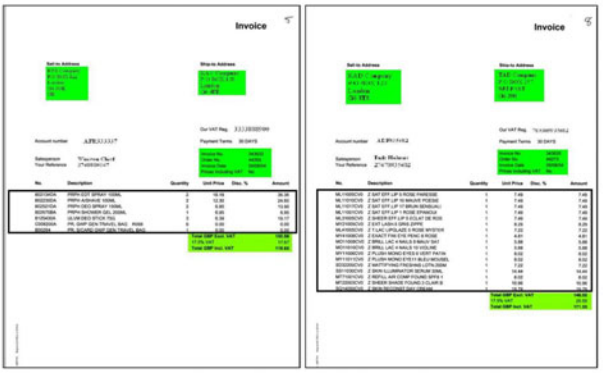

Fig. 5 Two documents from the same class

Figure 6 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.

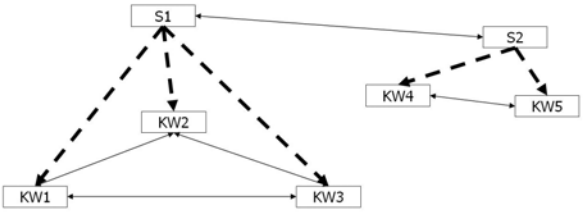

Fig. 6 A document model

4.3 Problem Solving

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 independent 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 4 shows the flow of this approach. Further details can be found in [43].

4.3.1 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 [48]. 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 $Pb2 = \sum |freq(E_{G1}) - freq(E_{G2})|$, 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.

4.3.2 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 following: 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 [44]), 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 [45]) 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.

4.3.3 Incremental Growing Neural Gas

The IGNG is an improvement of the GNG in some aspects (algorithm explained below). As shown in [47], 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 datum 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.

4.3.4 IGNG Improvement

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

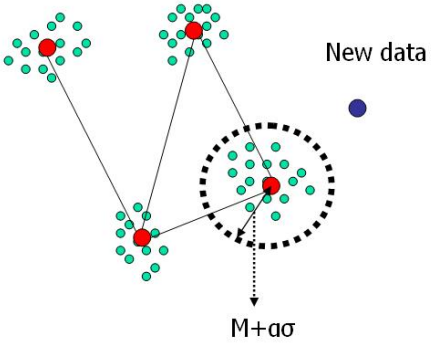

Fig. 7 A new data creates an embryon neuron

4.3.5 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 8. 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.

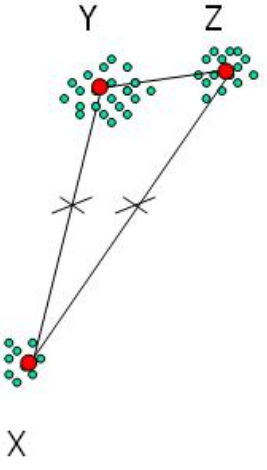

Fig. 8 A neuron that should not be deleted, even if it is not connected to any other neuron

4.3.6 Adaptation to the Case of Graph Classification

In [46], 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. Every 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 $\varepsilon(X,Y)$, ε being a real quantity. The same operation in the graph domain means that G1 has to be modified by $\beta = \varepsilon \cdot d(G1,G2)$ (equivalent to $Neuron = \varepsilon_b.(Neuron_{nearest} - entry)$). 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 [46].

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 second experiments are only related to incremental learning using administrative documents.

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 [43].

5.2 Experiments on Administrative Documents

Our experiments were performed 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.

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 1 gives these results.

Table 1 Influence of α and a_{edge}

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%

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 1, 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 performance 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.

We have demonstrated that the Case-Based Reasoning approach is natively adapted to heterogeneous document flows because each document is a case. The aim of the system is to organize and retrieve the different cases from an incremental knowledge base and moreover to adapt the previous knowledge to cope with a new document. The major concept of the adaptation approach comes from the observation that, specifically in the invoicing domain, the data (address, invoice date, amount) are often presented in few similar ways, in spite of each supplier editing and printing the invoices by its own feeling. Therefore, the 2-scales reasoning Global and Local approach demonstrates how it is possible to benefit from the redundancy of local structures to automatically model a whole new document.

This adaptation feature should reduce the time spent on modeling, which is one of the most important issues of Administration Documents Processing systems in the face of a large heterogeneous document flow.

In addition to invoices, this system can be applied directly to a very large majority of administrative documents in all domains: insurance, health, banking. We should recall that document processing targets the extraction of a finite set of indexes. These indexes are framed by business definitions obviously shared by the writer and the reader, otherwise there is no communication. Each writer can be free to organize the information set within a document and then introduce a large variety of documents. But the various ways to represent an index may converge, depending the accuracy of the business definition. Very formalized indexes will be very redundant (social security number for instance) and, on the other hand, indexes explained in a natural language text will support a batch of representation (that cannot be modeled by a structure).

Another perspective is to generalize the multi-scale CBR approach to manage the document flow structure. So far we have dealt only with documents, and supposed that the system input is the document. This supposition is true from a functional point of view. The end user handles each document separately and may define a data-extraction scope for each document. This is why all document analysis systems use the document scale as the system input scale. In fact the document is not the input scale but the output scale. The real input of the system is the result of the capture process: single page images. Using a fax machine or a desktop scanner, the user can digitize documents one by one. This makes sense when the user digitizes one or two documents, it makes no sense when production scanners which capture 1 000 to 10 000 pages per hour are used. Here the paper documents are turned into a flat sequence of page images. The challenge is then to segment the flat sequence into a structured document flow. The easiest solution is to introduce document separators (white sheets for instance) when preparing the batches to capture. This takes time and is paper-consuming. The other solution is document analysis to identify certain master documents in order to trigger the flow segmentation. For instance, a document starts at each page nb. 1 of an invoice or at each cheque. This solution supposes modeling of the flow sequence and the ability to identify before segmenting. To complete the complexity we should note here that the flow structure can be a multi-level tree: for instance the end users payment business document is a complex document composed of two more natural documents that are one (multi-page) invoice plus one cheque. We believe that we can expand the multi-scale CBR system in adding new CBR cycles to model the whole flow. We expect in this very global system to take advantage of redundancy in the documents and sub-structures at each level of the tree to enlarge the system scope and minimize the end users conception efforts.

Introducing the Case-Based Reasoning combined with Optical Character Recognition and Document Structure representation is a new step from Vision to Artificial Intelligence. For the last 40 years, the biggest challenge was to interpret the pixel into more symbolic information: characters, words, tables introducing more and more language models and semantic information to drive recognition. This pure recognition challenge is far from being solved, but the solution has reached a good enough level of quality to move slowly away from the document analysis scope towards a more ambitious document understanding scope. We now do not just ask the computer to read information and follow a fixed analysis process, defined by a grammar or a technical template. We can now ask the future system to simply learn from samples, to build its own rules, to infer new knowledge and to memorize enough of it so that human assistance is no longer needed.

If you will allow us the following comparison: the document reader system was yesterday a little boy able to recognize some patterns. He can today re-use part of his experience and learn himself from read documents in order to infer all this information onto new documents. On our side we will continue to raise our little boy and we wish you to conceive many brothers.

References

1. Grosicki, E., Carr, M., Brodin, J.-M., Geoffrois, E.: Results of the RIMES Evaluation Campaign for Handwritten Mail Processing. In: Int. Conf. on Document Analysis and Recognition (ICDAR) (2009)
2. Grosicki, E., El Abed, H.: ICDAR 2009 Handwriting Recognition Competition. In: 10th Int. Conf. on Document Analysis and Recognition, ICDAR (2009)
3. Bunke, H.: Recognition of cursive Roman handwriting - past, present and future. In: Int. Conf. on Document Analysis and Recognition (ICDAR 2003), vol. 1, pp. 448–459 (2003)
4. Lladós, J., Valveny, E., Sánchez, G., Martí, E.: Symbol recognition: Current advances and perspectives. In: Blostein, D., Kwon, Y.-B. (eds.) GREC 2001. LNCS, vol. 2390, pp. 104–127. Springer, Heidelberg (2002)
5. Chang, M., Chen, S.: Deformed trademark retrieval based on 2d pseudo-hidden markov model. Pattern Recognition 34, 953–967 (2001)
6. Tombre, K., Tabbone, S., Dosch, P.: Musings on symbol recognition. In: Liu, W., Lladós, J. (eds.) GREC 2005. LNCS, vol. 3926, pp. 23–34. Springer, Heidelberg (2006)
7. Krishnamoorthy: Syntactic segmentation and labeling of digitalized pages from technical journals. PAMI (1993)
8. Yamashita, A., Amano, T., Takahashi, I., Toyokawa, K.: A model based layout understanding method for the document recognition system. In: Int. Conf. on Document Analysis and Recognition 2003, ICDAR (1991)
9. Duygulu, P., Atalay, V.: A hierarchical representation of form documents for identification and retrieval. Int. Journal on Document Analysis and Recognition (IJDAR) 5(1), 17–27 (2002)
10. Mao, J., Abayan, M., Mohiuddin, K.: A model-based form processing sub-system. In: Int. Conf. on Pattern Recognition, ICPR (1996)
11. Sako, H., Seki, M., Furukawa, N., Ikeda, H., Imaizumi, A.: Form reading based on form-type identification and form-data recognition. In: Int. Conf. on Document Analysis and Recognition 2003 (ICDAR), Scotland (2003)
12. Ting, A., Leung, M.K.H.: Business form classification using strings. In: 13th International Conference on Pattern Recognition (ICPR), p. 690. IEEE Computer Society, Washington, DC, USA (1996)
13. Hroux, P., Diana, S., Ribert, A., Trupin, E.: Etude de methodes de classification pour l'identification automatique de classes de formulaires. In: Int. Francophone Conferenceon Writing and Document Analysis, CIFED (1998)
14. Ishitani, Y.: Model based information extraction and its application to document Images. In: Int. Workshop on Digital Library and Image Analysis, DLIA (2001)
15. Cesarini, F., Gori, M., Marinai, S., Soda, G.: Informys: A flexible invoice-like form-reader system. IEEE Trans. Pattern Anal. Mach. Intell. 20(7), 730–745 (1998)
16. Belad, A., Belad, Y., Valverde, L.N., Kebairi, S.: Adaptive Technology for Mail-Order Form Segmentation. In: Int. Conf. on Document Analysis and Recognition (ICDAR), Seattle, USA, pp. 689–693 (2001)
17. Wahl, F., Wong, K., Casey, R.: Block segmentation and text extraction in mixed text/image documents. Graphical Models and Image Processing 20 (1982)
18. Nagy, G., Seth, S., Viswanathan, M.: A prototype document image analysis system for technical journals. Computer 25 (1992)
19. Pavlidis, T., Zhou, J.: Page segmentation and classification. Graphical Models and Image Processing 54 (1992)

20. Sako, H., Seki, M., Furukawa, N., Ikeda, H., Imaizumi, A.: Form reading based on form-type identification and form-data recognition. In: Int. Conf. on Document Analysis and Recognition (ICDAR), Scotland (2003)
21. Laroum, S., Bchet, N., Roche, M., Hamza, H.: Hybred: An OCR document representation for classification tasks. International Journal on Data Engineering and Management (2009)
22. Zhong, S.: Efficient online spherical k-means clustering. In: Proceedings IEEE of the International Joint Conference on Neural Networks, IJCNN 2005, Montreal, Canada, July 30 - August 4, pp. 3180–3185 (2005)
23. Vapnik, V., Chervonenkis, A.: A note on one class of perceptrons. Automation and Remote Control 25 (1964); SVM & Boosting
24. Bartlett, P., Shawe-Taylor, J.: Generalization performance of support vector machines and other pattern classifiers. In: Scholkopf, B., Burges, C.J.C., Smola, A.J. (eds.) Advances in Kernel Methods Support Vector Learning, pp. 43–54. MIT Press, Cambridge (1999)
25. Kim, J., Le, D.X., Thoma, G.R.: Automated labeling in document images. In: Document Recognition and Retrieval VIII (2001)
26. Cesarini, F., Marinai, S., Sarti, L., Soda, G.: Trainable table location in document images. In: Int. Conf. on Pattern Recognition (ICPR), vol. 3, pp. 236–240 (2002)
27. Coüasnon, B.: Dealing with noise in DMOS, a generic method for structured document recognition: An example on a complete grammar. In: Lladós, J., Kwon, Y.-B. (eds.) GREC 2003. LNCS, vol. 3088, pp. 38–49. Springer, Heidelberg (2004)
28. Coasnon, B.: Dmos, "a generic document recognition method: application to table structure analysis in a general and in a specific way. Int. Journal on Document Analysis and Recognition 8(2-3), 111–122 (2006)
29. Conway, A.: Page grammars and page parsing: A syntatic approach to document layout recognition. In: Int. Conf. on Document Analysis and Recognition, ICDAR (1993)
30. Niyogi, D., Srihari, S.N.: Knowledge-based derivation of document logical structure. In: Int. Conf. on Document Analysis and Recognition, ICDAR (1995)
31. Dengel, A., Dubiel, F.: Computer understanding of document structure. IJIST (1996)
32. Amano, A., Asada, N.: Graph Grammar Based Analysis System of Complex Table Form Document. In: Int. Conf. on Document Analysis and Recognition (ICDAR) (2003)
33. Sainz Palmero, G.I., Cano Izquierdo, J.M., Dimitriadis, Y.A., Lopez, J.: A new neuro-fuzzy system for logical labeling of documents. Pattern Recognition (1996)
34. LeBourgeois, F., Souafi-Bensafi, S., Duong, J., Parizeau, M., Cotc, M., Emptoz, H.: Using statistical models in document images understanding. In: DLIA (2001)
35. Rangoni, Y., Belad, A.: Data Categorization for a Context Return Applied to Logical Document Structure Recognition. In: Int. Conf. on Document Analysis and Recognition (ICDAR) (2005)
36. Rangoni, Y., Belad, A.: Data Categorization for a Context Return Applied to Logical Document Structure Recognition. In: Int. Conf. on Document Analysis and Recognition, ICDAR (2005)
37. Sainz Palmero, G.I., Cano Izquierdo, J.M., Dimitriadis, Y.A., Lopez, J.: A new neuro-fuzzy system for logical labeling of documents. Pattern Recognition (1996)
38. Hamza, H., Belaïd, Y., Belaïd, A.: Case-based reasoning for invoice analysis and recognition. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 404–418. Springer, Heidelberg (2007)
39. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. IOS Press, Amsterdam (1994)

40. Burke, R., Hammond, K., Kozlovsky, J.: Knowledge-based information retrieval from semistructured text (1995)
41. Kolodner, J.: Maintaining organization in a dynamic long-term memory. Cognitive Science (1983)
42. Watson, I., Marir, F.: Case-based reasoning: A review 9, 355–381 (1994)
43. Hamza, H., Belaïd, Y., Belaïd, A.: Case-based reasoning for invoice analysis and recognition. In: Weber, R.O., Richter, M.M. (eds.) ICCBR 2007. LNCS (LNAI), vol. 4626, pp. 404–418. Springer, Heidelberg (2007)
44. Fritzke, B.: Growing cell structuresa self-organizing network for unsupervised and supervised learning. Neural Networks 7(9), 1441–1460 (1994)
45. Hodge, V.J., Austin, J.: Hierarchical growing cell structures: Treegcs. Knowledge and Data Engineering 13(2), 207–218 (2001)
46. Gunter, S., Bunke, H.: Self-organizing map for clustering in the graph domain. Pattern Recognition Letters 23(4), 405–417 (2002)
47. Prudent, Y., Ennaji, A.: A new learning algorithm for incremental self-organizing maps. In: ESANN, p. 712 (2005)
48. Lopresti, D.P., Wilfong, G.T.: A fast technique for comparing graph representations with applications to performance evaluation. IJDAR 6(4), 219–229 (2003)