A case-based reasoning approach for invoice structure extraction

Hatem Hamza, Yolande Belaïd, Abdel Belaïd

A case-based reasoning approach for invoice structure extraction

Hatem Hamza [*,**], Yolande Belaïd[**], Abdel Belaïd[**]
[*]ITESOFT, France
[**]LORIA, University Nancy 2. France
{hamza,ybelaid,abelaid}@loria.fr

Abstract

This paper shows the use of case-based reasoning (CBR) for invoice structure extraction and analysis. This method, called CBR-DIA (CBR for Document Invoice Analysis), is adaptive and does not need any previous training. It analyses a document by retrieving and analysing similar documents or elements of documents (cases) stored in a database. The retrieval step is performed thanks to graph comparison techniques like graph probing and edit distance. The analysis step is done thanks to the information found in the nearest retrieved cases. Applied on 950 invoices, CBR-DIA reaches a recognition rate of 85.29% for documents of known classes and 76.33% for documents of unknown classes.

1. Introduction

Industrial invoice analysis systems require a continuous adaptation capacity to the structure variation of the documents. They have to deal with documents having very different structures. They have to analyse two different types of information: key-words and tables. Table extraction and understanding has been a subject of interest in the last years. Many approaches were proposed in the litterature. In [6], a simple method for table detection was proposed. It is based on the analysis of the gap between table fields in an image (this gap is supposed to be larger than the gap betwwen words in a text line). This method reaches a detection rate of 97.21%. Some other works rely on the data extracted from the image (words, text) to detect and interpret tables. In [2], a morphologiclal approach for table fields tagging was proposed. By analysing the nature of each word in the table zone, each field is given an attribute (an interpretation: "total amount", "code"...). However, it was applied on tables that were already extracted. A very good survey about table extraction and understanding can be found in [4]. Invoice and form processing tackles a number of other research topics. Invoice analysis using key-words was proposed in [7] [3]. Both of these papers showed good results. However, they were limited to processing isolated key-words not taking into account the context of each key-word. Other works are interested in document classification using graphs of key-words [8]. In order to have a complete system for invoice analysis, we need all of the previous ideas. CBR-DIA works on all these problems. It processes invoices of both known and unknown classes. It analyses information related to key-words and tables. The main idea of our approach is to use not only the information on the processed document but also the knowledge and experience acquired while analysing other documents.

Inspired by what AI literature proposed in problem resolution techniques, we propose in this work to experiment CBR [1] for invoice structure extraction. The developed CBR-DIA approach tackles three main problems: structure extraction, structure and document similarity search and new structures analysis and interpretation. The paper is organized as the following: section 2 introduces briefly CBR and its main concepts. Sections 3, 4 and 5 present CBR-DIA's architecture. Section 6 shows the obtained results and their interpretation.

2. Case-based reasoning

CBR is a powerful problem solving strategy that uses previous experiences to process new problems that have not been processed before [1]. The "Problem" is the input of any CBR system. It is the first component of a case in the CBR terminology, a case being the set (problem, solution). Its resolution (to find the solution) consists in three main phases:

1. similar case retrieval from the database ("Search");
2. adaptation of the solution to the studied problem ;
3. learning of new solved cases ("Training").

In CBR-DIA, two sorts of cases are defined: a document case and a structure case. The flow of CBR-DIA as shown

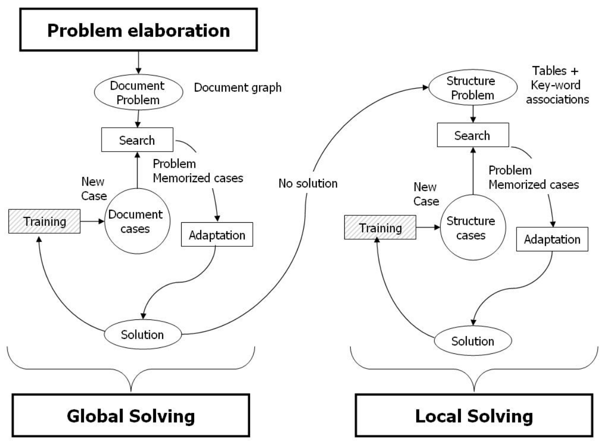


Figure 1. CBRDIA flow

in figure 1 is based on three main steps: problem elaboration, global problem solving and local problem solving. Problem elaboration consists in information extraction from the document. Invoices contain many information that need to be analysed and interpreted: key-words, table rows. The extraction step is called problem elaboration in CBR terminology. In CBR-DIA, key-words and tables represent the problem as they are the simplest information to be extracted. They require however to be analysed, if we want to understand what is behind these information. For example, it is useless to know that an invoice contains a table with 3 rows and 5 columns if we do not know what is the information contained in this table.

The problem is then solved using either global solving or local solving. The former is used when a similar document case exists in the database (i.e. Document cases). The latter is used when the processed document can not be classified and is completely new. In that case, the problem is solved by analysing its elementary structures one by one.

3. Problem elaboration

The system input is a raw document given by OCR. The OCR file contains the list of words and coordinates. The document is represented by the set of words: $W_i, i = 1..n$.

3.1. Data extraction and coding

The first step consists in re-organizing the words in a more logical way. First, each word is given three attributes: position, key-word and nature. The attribute "nature" is represented by an alphabetical character: for example, 'A' for numerical, 'B' for alphabetical, etc. A word is tagged as a key-word if it belongs to a predefinite list of key-words. These key-words are words that occur frequently in administrative documents. They can be in several languages. The list of key-words is updated regularly.

Then, fields are constituted by gathering neighbour words horizontally. Each successive pair of words (Wi, Wj) in a field verifies $d(Wi, Wj) < \delta$ where δ is a threshold depending on the character size of the field words. A field is characterized by two attributes: position and nature. The nature of a field is deduced from its words' natures. For example, if a field contains an alphabetical and a numerical word, then it will be tagged 'C' for alphanumerical.

From fields, we extract horizontal lines and vertical blocks. Fields' neighbourhoods and alignments are used to constitute these lines and blocks. A vertical block is a set of fields vertically aligned. Two vertical fields Fi and Fj are in the same vertical block if $d(Fi, Fj) < \beta$ where β is a threshold depending on the fields size and position. Similarly, we use a threshold for horizontal fields. A line or a block have the following attributes: position and pattern. A pattern is string composed of fields' tags list. For example, if the fields in the line have the tags: 'A', 'B', 'B' and 'C', then the pattern is "ABBC". These patterns will be used in table extraction. After these elementary information are extracted, high level structures are extracted. They can be either pattern structures (PS) when related to tables or key-word structures (KWS) when related to local arrangements of key-words. Figure 2 shows a document containing 4 KWS and a PS. The KWS are in gray-tone, whereas the PS is the bold box.

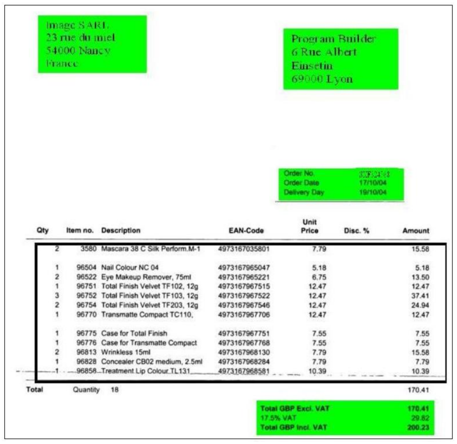


Image SARL
23 rue du miel
54000 Nancy
France

Program Builder
6 Rue Albert
Einsetin
69000 Lyon

Order No.	[illegible]
Order Date	17/10/04
Delivery Day	19/10/04

Qty	Item no.	Description	EAN-Code	Unit Price	Disc. %	Amount
2	3580	Mascara 38 C Silk Perform.M-1	4973167035801	7.79		15.58
1	96504	Nail Colour NC 04	4973167965047	5.18		5.18
2	96522	Eye Makeup Remover, 75ml	4973167965221	6.75		13.50
1	96751	Total Finish Velvet TF102, 12g	4973167967515	12.47		12.47
3	96752	Total Finish Velvet TF103, 12g	4973167967522	12.47		37.41
2	96754	Total Finish Velvet TF203, 12g	4973167967546	12.47		24.94
1	96770	Transmatte Compact TC110,	4973167967706	12.47		12.47
1	96775	Case for Total Finish	4973167967751	7.55		7.55
1	96776	Case for Transmatte Compact	4973167967768	7.55		7.55
2	96813	Wrinkless 15ml	4973167968130	7.79		15.58
1	96828	Concealer CB02 medium, 2.5ml	4973167968284	7.79		7.79
1	96858	Treatment Lip Colour TL131	4973167968581	10.39		10.39
Total	Quantity 18					170.41

Total GBP Excl. VAT	170.41
17.5% VAT	29.82
Total GBP Incl. VAT	200.23

Figure 2. An invoice containing 4 KWS and a PS

3.2. PS extraction

PS are consecutive horizontal lines having similar patterns. This is the case of a table. Figure 2 shows a document containing a PS composed of 12 horizontal lines having the pattern "AACAAA". This means that there are two numerical columns, one alphabetical column and four other numerical columns.

The PS extraction process contains three steps:

- For each horizontal line, a list of neighbour lines HLN is constitued using edit distance on their strings (i.e. patterns). We use a threshold (usually equal to 1 in order to accept only 1 transformation between strings) between line patterns to find neighbours;

- The list of each group of neighbour lines is studied based on the fields' positions. In figure 3, the edit distance between the patterns is null, as they represent the same string "ABCC". However they do not correspond to the same PS because of the difference of the spatial positions. To avoid such confusions when the edit distance is null, we take into account patterns' fields positions as the following. For every list HLN we compute a new matching value. This value depends on the number of exact vertical alignment of fields having the same tag. The final matching value is the ratio in (1):

$$RT = \frac{|matching\ fields|}{|fields\ in\ HLN|}. \qquad (1)$$

  The higher RT is (RT tends to 1), the more probable HLN is a PS. If $RT = 1$, HLN is a singleton (this case will be eliminated because it is meaningless for table) or HLN is a perfect table.

A	**B**	**C**	**C**	
A	**B**		**C**	**C**

Figure 3. Two patterns with edit distance=0

- After processing the whole document, the chosen HLN is the one maximizing RT. PS is then the best HLN candidate. This method can extract tables only when there are at least two table lines in the document.

3.3 KWS extraction

KWS are constitued from neighbour KW like "road", "zip-code", "name" for an address. KWS are very important in invoices as many details are expressed in such structures. We use graphs to represent KWS (key-words in vertices, and spatial relationships on edges). KWS maintain the spatial relationships as well as the semantic proximity between KW.

3.4 Document graph extraction

A document graph is composed of its structuring elements (PS and KWS). The vertices are either PS or KWS. The edges link the different structures. In this graph, two sorts of edges exist: "spatial" (left, right, top, bottom) when they designate spatial relationships or "contain" when they designate a structure component (as between a KWS and each of its KW). This kind of graph representation gives flexibility to CBR-DIA as it is just articulated around relative structure positions. It is also helpful for document classification while using graph probing(see 4.1).

3.5 CBR-DIA cases

CBR requires the definition of cases: a problem and its corresponding solution. According to the problem elaboration step, three different cases are possible:

1. KWS case: the problem is the graph of key-words contained in a structure. The solution is the interpretation of each KW. For example, the solution of the KW "street" is the name of the street and the number corresponding to the address (12 Decker street). In this case, KWS solution is the set of KW solutions;

2. PS case: the problem is the pattern (e.g "ABBB") representing the table and the solution is the interpretation of each table column;

3. Document case: the problem is the document graph and the solution is the solution of all its structures.

4. First CBR cycle: global solving

After extracting the problem from the given document, a solution should be extracted from the document. Global solving consists first in checking if a similar case exists in the document database. This is done by measuring a distance between the document graph and those of the cases stored in the document database.

4.1. Similar case retrieval

Graph probing is used for this purpose [5]. It is a fast and accurate technique of graph comparison. It measures the degree of dissimilarity of two graphs G_1 and G_2. In [5], Lopresti applied his method successfully on document graphs containing simple structures of lines and words. Two probes have to be measured: a probe on vertices P_1, and another one on edges P_2. P_1 is simply calculated by measuring the vertices' frequencies in each graph. The edge structure of each vertex is represented by a four-tuple Y=(top, left, right, bottom). The frequency of each direction is reported in this four-tuple for every vertex. The frequency of each Y is calculated in each graph producing P_2. The final probe is:

$$P(G_1, G_2) = (P_1(G_1) - P_1(G_2)) + (P_2(G_1) - P_2(G_2))$$

It has to be noticed that if the final probe is null, G1 and G2 are not necessarily isomorphic. However, graph probing gives an approximation of the edit distance. Graph probing is applied in CBR-DIA on each document to find the most similar document case in the document database.

4.2. Solution adaptation

When a similar case is found, the adaptation consists first in finding for each structure in the graph of the problem (G1) the corresponding structure in the database graph (G2). As G1 and G2 correspond to the same case (they belong to the same set of documents), the system just copies the information about the nature (alphabetical, numerical) and the position (left on the same line, right on the same line, top on the line, above) of each solution of G2 and looks for similar information in G1. For example, if the solution corresponding to a KW "total" in G2 case has the properties "real number + right", the system will look for a real number on the right of "total" on the same line in G1. If an answer exists, then it is proposed as a solution for this KW.

5. Second CBR cycle: local solving

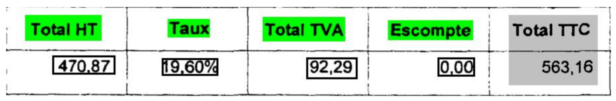
If no similar case exists in the document database, the system builds a solution based on the structures already processed in others documents and stored in a structure database.

5.1. KWS solving

The solving procedure acts as the following:

For each KWS in the document, the system looks for the nearest structure in the structure database and adapts its solution to the processed KWS. Graph edit distance is used to find the nearest graphs in the database. We used edit distance between these graphs as we are really looking for graph isomorphism, or at least, sub-graph isomorphism. As KWS graphs are also small (no more than 5 vertices per graph in general), it is then better to use a more precise comparison technique than to use a faster but less accurate one like graph probing distance. The cost function used to compute edit distance between graphs has uniform costs for both vertices and edges edit operations as both KW and their relative positions seem to have the same importance in the graph.

The nearest structures' solutions are now adapted to the document structures. As the cases in the database have already a correct solution, its adaptation consists in taking the solution of each KW (case of KWS) and trying to find a corresponding solution in the processed document. If a complete solution can not be found, the following processing has to be done. For KW, some universal knowledge exists and it would be really a waste of time not to take advantage of it. For example, it is usual that the KW "total" is followed by a numerical. This numerical can be a real number or an integer depending on the document but its numerical nature is always valid. A rule basis detailing the general rules associated with key-words was built to complete any partial solution of a KWS. This basis allows completing some missing KWS solutions. It has to be noticed that this rule basis is not able to solve complete cases as it does not take into account the context of the structure. Moreover, the rule basis knowledge is very general and is not related to any concrete case. The example in Figure 4 shows a KWS which nearest KWS in the structure database solves four out of five KW. By using a rule basis, a complete solution can be found.



Total HT	Taux	Total TVA	Escompte	Total TTC
470,87	19,60%	92,29	0,00	563,16

Figure 4. A KWS. Only the KW Total is solved by the rule basis.

5.2. PS solving

Each extracted PS is compared with the database cases to retrieve the nearest structure. As PS are represented with strings, their patterns are compared using string edit distance. When a similar PS is found (same pattern, or with a maximum of one transformation), the table columns of the extracted case are given the tags of the database case, unless the rule between the fields of the tables do not match. In this case, the system tries to find the rule between extracted fields by trying the rules in other close PS cases (close PS cases with more than one transformation) until a valid rule is found.

6. Experiments

Our approach was tested on 950 documents. They are divided in 2 groups:

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

The results are described thanks to three different measures (2) where X can be a document, a KWS or a PS.

$$R_X = \frac{|correct\ solutions\ |}{|solutions\ in\ ground\ truth\ X|}. \tag{2}$$

A correct solution corresponds to a KW's solution or to a field in a PS that has been correctly extracted and interpreted. The structure database contained initially 300 structures. Only 20% of the tested structures have a complete similar case in the database. The remaining cases are taken from several other documents which are not related to the tested documents. We chose to test our system in this way to show its ability to find a solution for a given problem even if it has never been studied before.

The results are given in table 1. They are satisfying from an industrial point of view.

Table 1. Results of CBR-DIA for global solving and local solving

	R_{doc}	R_{kws}	R_{ps}
Global Solving	85.29%	82.22%	88.75%
Local Solving	76.33%	76.38%	76.28%

As expected, global solving reaches better results that local solving. In fact, global solving uses a previous complete solution of a document. Solving a new case means copying the solution of an old one. On the other hand, local solving looks for a document solution structure by structure. Local solving will then ignore the whole context of the document by trying to analyse it part by part.

In global solving, the missing 16.63% correspond to 7.76% system errors and 7.17% OCR errors.

In KWS local solving, errors are due to:

- 16.57% of system errors (bad solution, no solution found, confusion with other solutions);
- 8.08% of OCR errors.

In PS local solving, errors are due to:

- 16.66% of bad detection of table lines or a bad proposed solution (missing lines, no detection of table);
- 7.14% of OCR and segmentation errors (for example, the word 23.7 is read as 23.T, two fields are fused together).

The OCR used in this application is a professional one used by ITESOFT. OCR errors are not just due to the software performance, but they also depend on:

- the quality of documents. In our dataset, we had about 8% of documents of very poor quality (this can be caused by the original quality of the document, or by a bad scanning);
- noisy information such as missing characters.

A special case of KWS was also tested (addresses). We tested KWS solving on 30 documents containing saddresses. 78.33% (118/150) of good results were obtained. We can notice that this special case exists not only in invoice documents, but also in any other administrative documents.

7. Conclusion and future works

A CBR approach for invoice document analysis and interpretation was proposed in this paper. CBR-DIA produces good results even if the documents have never been processed by the system before. This work is still under study in several ways. We are studying the improvement of problem elaboration and especially of PS extraction. We are also focusing on database indexing in order to reduce the solving time. This is an important problem in CBR and in document analysis. In fact, the more CBR-DIA is used, the more cases (documents and structures) databases will contain. Allowing a fast and accurate access to these bases is then very important for the system's performance.

References

[1] A. Aamodt and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. In *IOS press*, 1994.

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

[5] D. P. Lopresti and G. T. Wilfong. A fast technique for comparing graph representations with applications to performance evaluation. *IJDAR*, 6(4):219–229, 2004.

[6] S. Mandal, S. P. Chowdhury, A. K. Das, and B. Chanda. A simple and effective table detection system from document images. *IJDAR*, 8(2-3):172–182, 2006.

[7] H. Sako, M.Seki, N. Furukawa, H.Ikeda, and A.Imaizumi. Form reading based on form-type identification and form-data recognition. In *ICDAR*, Scotland, 2003.

[8] A. Schenker, M. Last, H. Bunke, and A. Kandel. Comparison of distance measures for graph-based clustering of documents. In *GbRPR*, pages 202–213, 2003.