A case based reasoning approach for invoice structure extraction

Abstract

This paper shows the use of Case Based Reasoning (CBR) in the extraction of structures in invoices. This method, named CBR-DIA (CBR for Document Invoice Analysis), is generic as it is class independent and does not need any previous training. Based on the accumulation of processing experiences on a variety of documents, and on real problems (word configurations) stemmed from the given document, this method solves the problem (i.e. find its solution) by retrieving the nearest case (processing experience) in the database. Problems are represented by graphs and strings. The retrieval step is performed using graph comparison techniques based on probe and edit distances. Applied on 923 invoices, we reach a recognition rate 85,22% for documents of known classes and 74,90% for unknown classes.

1. Introduction

Form analysis systems in the real production chains are often faced to a huge quantity of documents requiring a high processing velocity and a continuous adaptation capacity to the structure variation. The manual and even semi-automatic solutions which consist in adapting the structure model for each set of new documents can no long be used because of the heavy modeling phase they require. Bunke [bunke] propose a similar approach by creating a manual pattern for each specific class of documents. He uses a matching technique preceded by image preprocessing steps for scaling, deskewing, etc.

The most promising approach is the one developed by Cesarini [ces] for invoices. Limited to some keywords, the corresponding structures are interpreted by combining two levels of knowledge: intra- and inter-classes knowledge. Starting from key-words extracted from a document, the system looks for the solution first using the intra-class knowledge (for ex. Global coordinates, tags, etc.) and in failure case it exploits the inter-classes knowledge (a set of intersections of class information as for ex. “total + Numeric ” coming from the “ total+ real” coming from one class and “total + integer” coming from another class)). The application of this approach is limited to isolated keywords not taking into account more frequent and important structures in forms such as tables, addresses, etc.

Inspired by what proposed in the literature concerning the problem resolution techniques in AI like the case-based **diagnostic** systems, we propose in this work to experiment CBR for invoice structure extraction. The developed CBR-DIA approach tackles three main problems: dynamic structure modeling, structure and document similarity search, and new structures discovery and interpretation.

The paper is organized as the following: the first section introduces briefly CBR and its main concepts. The second section presents our system’s architecture. The third section shows the obtained results and their interpretation. Finally, some short term and long term perspectives are given.

2. Case Based Reasoning

the general CBR flowchart is shown in Figure 1.

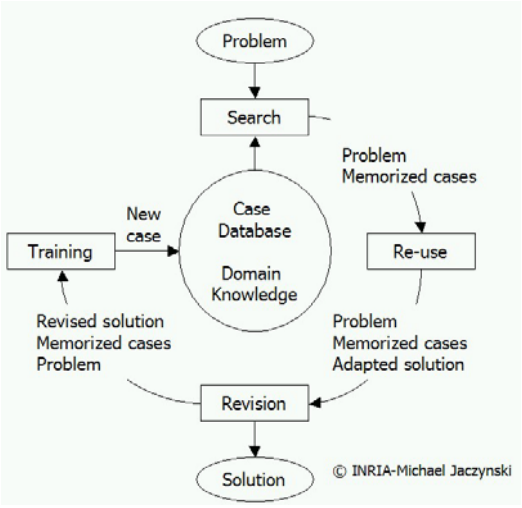

Figure 1: CBR flow

The ‘Problem’ is the input of any CBR system. It is the first component of a case in the CBR terminology (a case c={problem, solution}). Its resolution (to find the solution) consists in three main phases: 1) similar case retrieval from the database (‘Search’) , 2) The solution adaptation to the studied problem: ‘Re-use’ + ‘Revision’, and 3) learning of new solved cases: ‘Training’.

In our approach, we define two sorts of cases that will be detailed further in section 2.3.2: a document case and a structure case. As a structure is a part of a document, it becomes obvious that a structure case is

always included in a document case. This implies directly that solving a document case implies necessarily solving structure cases.

The flow of our approach, as shown in Figure 2 as shown in is based on three main steps: problem elaboration, global problem solving and local problem solving.
Problem elaboration consists in indices extraction from the document. These indices are either key-words (i.e. stemmed from domain knowledge) and their spatial relationships, or table rows. This problem is then solved using either global solving process or local solving one. The former is used once a similar document case exists in the database (i.e. Document cases). The latter is used when the processed document is completely new. In that case, the problem is solved by solving its elementary structures one by one.

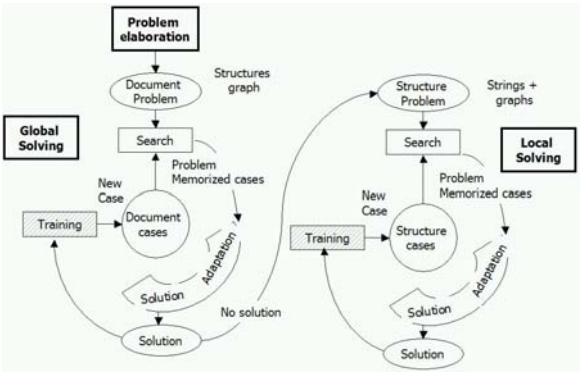

Figure 2: CBR-DIA flow

3. Problem elaboration

The system input is a raw document given by OCR. The OCR file written in XML contains the list of words and coordinates. The document is represent by the set $D=\{W_i\}I=1,..,n$.

3.1 Data extraction and coding

The first step consists in re-organizing the given list of document words d in a more logical and organized way.
First, each word W is characterized by a list of attributes:

- position
- KW: key word or not
- nature: represented by an alphabetic character: for ex. A for numerical, B for alphabetical, etc.

This information is helpful for the future coding steps.

Then, fields (F_j) are constituted from D by gathering neighbour words horizontally. Each successive pair of words (W_i, W_j) in F verifies the property: $d(W_i, W_j) \leq \delta$ where δ is a threshold depending on the character size of the field words. F is also characterized by a list of attributes: position, nature. Rules are used for characterising this nature from those of the words.

From fields, we extract horizontal and vertical lines (HL, VL). As for fields, we use neighbourhood and the field alignments to constitute HL and VL. Then, HL and VL are used to extract the document structures.

A vertical line VH is a set of fields F vertically aligned. Two vertical filelds F_i and F_j are in the same vertical line if $d(F_i, F_j) \leq \beta$ where β is a threshold depending on the field size of the line. Similarely, we use the threshold α for horizontal fields. The line has the following attributes: position and pattern. A pattern is string composed of field tag list. For example, if the fields in the line have the tags: A, B, B and C, then the pattern is ABBC.

Once, these elementary information is extracted, this gives the basis for the extraction or high level structures (S) called either pattern structures (PS) when related to tables or key word structures (KWS) when related to local and non necessarily non-linear arrangements.

3.2 Structures extraction

Two structures are extracted from the documents, PS and KWS. We shall explain the extraction method in the following.

PS extraction

PS are composed of a list of consecutive HLs having similar patters. This is the case of a table. Figure 3 shows a PS composed of 4 HLs having the pattern ABABAAA.

5	Prix km jour Tarif C	147	Km	7.86	1 155.42	1
8	Petits bagages	10	Bg	2.10	21.00	1
7	Attente par 1/4 Heure	4	Unité	16.37	65.48	1
2	Prise en charge MINIBUS	1	Unité	180.00	180.00	1

Figure 3: Pattern based Structure (PS)

The PS detection process contains many steps:

First, for each HL, we constitute a list of HL neighbours HLN using edit distance on their strings (i.e. patterns). We use the threshold δ1 (usually equal to 1 or 2) between HL patterns.

The list of each HL neighbours studied and considered or not as a valid PS.

In Figure 4, the edit distance between the patterns is 0, as the represent the same string, however they do not correspond to the same PS because of the difference of the spatial positions.

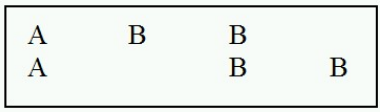

Figure 4: Two strings with edit distance = 0

To avoid such confusions when the editing distance is null, we take into account additional pattern fields positions as the following:

For every set HLN={HL fields} we compute a new matching value. This value depends on the number of exact vertical alignment of similar tags. The final matching value is the ratio R=(number of matching fields)/(total number of fields in HLN). R is always smaller than 1. The higher R is, the more probable HLN is a PS. In fact, if R is near 1, then two possibilities exist:

- R=1, HLN is a singleton (This case will be eliminated because it is meaningless for table)
- R<1, HLN is not a singleton, meaning the case of a possible table.

After processing the whole document, the retained HLN is the one maximizing R.

KWS extraction

KWS are local arrangements of keywords like 'road', 'zip-code', 'name', etc. for an address. These key-words are in several languages and occur frequently in even international administrative documents. We use graphs to represent this key-word association (key-words in vertices, and relationships on edges). This association maintain the spatial relationships as well as the semantic proximity between key-words.

KWS are extracted from VLs containing key-words.

3.3 Document structure extraction

A document structure is a gathering of all its sub-element structures (PS and KWS). We also use a graph for its representation. This graph is oriented and labelled containing 2 types of vertices: Structures (PS or KWS). The edges links the different structures.

In order to have a harmonious graph representation, useful for future comparisons, we consider all the vertices are visible at the same level. This means that the difference between vertices is characterized by edges which are either '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 provides a flexibility to CBR-DIA in the sense because it is just articulated around relative structure positions. A similar idea of document representation was introduced in [Suen icdar 95].

4. CBR-DIA Cases

CBR requires the definition of cases by defining for each one of them the problem and the corresponding solution. Let C be a case, C= {P_C, S_C}.

According to the problem elaboration step, three different cases are possible:

1/ **KWS case**: where P_KWS is the graph of key-words contained in a structure and S_KWS 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** *rue* **de la liberté**). In this case, KWS solution is the set of KW solutions. S_KWS = {S_KW} where KW is a particular case of KWS containing just one key-word.

3/ **PS case**: where P_PS is the pattern (e.g ABBB) representing the table and S_PS is the interpretation of each table column. We associate to S_PS mathematical rules connecting their patterns field. As an example, the total amount can be connected to others fields : 'unit' and 'number' by the following formulae: total = unit amount* number of units.

3/ Document **case (DC)**: P_DC is the document graph and S_DC is exactly the solution of all its structures: S_DC = {S_KWS, S_PS}.

4. Global Solving

After having extracted the problem P_DC from the given document, a solution S_DC should be extracted from the document DC. Global solving consists first in

checking whether a similar case exists in the Document cases database D_DB. This is done by measuring a distance between the graph P_DC and those of the cases stored in D_DB.

4.1. Similar case retrieval

For graph comparison, we use Lopresti's method called probing distance [ijdar 2003]. It is a fast and accurate technique to compare graphs by measuring their degree of dissimilarity. This method was applied successfully on document graphs containing simple structures of lines and words.

In order to compare labeled and directed graphs, two probes PB1 and PB2 have to be measured:
1/ PB1 = the frequency of each vertex in the graph;
2/ PB2 = the frequency of each vertex' edge structure.
Figure 5 shows a simple example of a graph of a document problem.

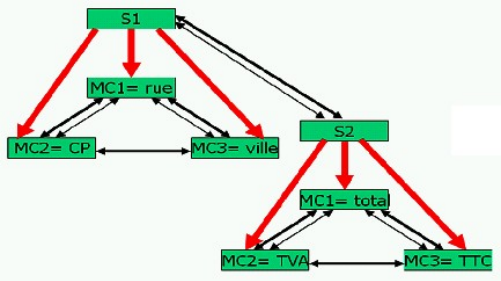

Figure 5: P-DC Example

Each document graph P_DC contains N vertices. A vertex can either be a structure node (labeled S) or a KW node. Each vertex has M edges connecting it to other vertices.

PB1 and PB2 are measured for both the studied graph P_DC (G1) and the D_DB graph (G2). The probing distance Pd between the graphs G1 and G2 is then :

Pd(G1,G2)=(PB1(G1)-PB1(G2))+(PB2(G1)-PB2(G2))

It has to be noticed that the probing distance is not a real distance between graphs. In fact, if Pd(G1,G2)=0, G1 and G2 are not necessarily isomorphic. However, it gives an approximation of the well known edit distance.

4.2 Solution adaptation

Once a similar case for P_DC (G1) is found in the D_DB (G2), the adaptation consists in finding for each structure in G1, the corresponding structure in G2. This is achieved easily by measuring the edit distance or the probing distance between the graphs of structures between G1 and G2.
As the documents correspond to the same case, the system just copies the information about the nature (alphabetical, numerical…) and the position (left, right, top…) of each solution and looks for similar information in the studied document.

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 the processed document. If an answer exists, then it is proposed as a solution for this KW.

5. Local Solving

If no similar case exists in D_DB, the system builds a solution based on the structures already processed in others documents and stored in a special database S_DB (Structure cases database).

5.1. KWS Solving

The building procedure acts as follows:

1/ For each structure in the document, find the nearest structure in the S_DB. The P_KWS graph is compared to the KWS cases of S_DB. The solution of the nearest structure is adapted.

Graph edit distance is used to find the nearest neighbors for structures. We have used edit distance between these graphs as we are really looking for graph isomorphism, or at least, subgraph isomorphism. As S_DB graphs are also small (no more than 5 vertices per graph in general), and as many graphs can be similar and just have some differences at the level of edges, it is the better to use a very precise comparison technique than to use a faster one like the graph probing distance.

2/ The nearest structures' solutions are now adapted to the document structures. As the cases in the SDB have already a correct solution, the 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 total solution is found for a structure, then, it can be stored in the S_DB. Otherwise, another processing has to be done.

For KW, some universal knowledge exists and it would be really a loss of time not to take advantage of

it. For example, it is usual that the KW "total" is followed by a numerical. Whether this numerical is a real number or an integer depends on the document, but this information is always valid.

Following this logic, a rule basis (RB) was built, in order to complete any partial solution of a KWS. RB helped us completing many KWS solutions (see results section).

The example in figure X shows a KWS whose nearest KWS in S_DB resolves three out of four KW. By using RB, a complete solution can be found.

////////////// figure

5.2. PS Solving

Each extracted PS is compared with the S_DB to retrieve the nearest structure. As PS are represented with strings, patterns are compared using string edit distance.
Once a close PS (C_PS) is found, two operations are performed on the studied PS (S_PS). First, the numerical rule of the C_PS is applied on S_PS. If C_PS rule works on S_PS (which means that numerical fields of S_PS have the same relationship than C_PC), then S_PS fields can have the same tags than those of C_PC.

For example, if C_PS rule is : "total amount = number of units * unit price", and numerical fields of S_PS are :

"12.00 5 60.00
4 52.20"

then the rule can be applied on S_PS. Moreover, as we know that "number of units" is generally an integer, and that "unit price" is smaller than "total amount", the interpretation of S_PS fields becomes immediate.

6. Results

Our approach is tested on 923 documents. They are divided in 2 groups: the first one contains 100 documents where each one has a similar case in D_DB: this will help us testing the global solving. The second one contains 823 documents for which no associated case exists in D_DB. Hence, local solving will be applied on these documents.

D_DB contains document cases. Each document case is written in an XML format containing both problem and solution. This XML document describes the structures in the document as well as the spatial relationships between them.

S_DB contains structure cases. All S_DB is stored in one XML format that can be enriched by new structures. As in D_DB, each structure case is described in details thanks to specific XML tags that we created for this purpose. Both problem and solution are described.

The results are the following:

Global solving		
Document number	Doc found	rate
…	….	85,22 %
Local Solving		
Structure number	Structure found	rate
….	…	74,90 %
Total zone		
Address zone		
Company ID zone		

The percentage corresponds to the ratio: number of right solutions)/ (number of desired solutions). A right solution corresponds to a KW solution or to a field in a PS that has been correctly extracted and interpreted.

In global solving, the missing 14.78% correspond to:

- 3.72 % system errors (see detailed results to understand system errors)
- 11.06% OCR errors.

Local solving is performed on KWS and PS. S_DB contains initially 300 structures. Only 20% of the tested structures have a complete similar case in SDB. The remaining cases in SDB are taken from several other documents which are not related to the tested documents. We have chosen 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 detailed results for KWS are:

Total KW	KW found	Rate
2263 KW	1703	75,25%

The errors are due to :

- 12… % system errors (bad solution, no solution found, confusion with other solutions)
- 12…. % OCR errors

PS fields	Found fields	Rate
7100	5310	74,80%

The errors are due to:

- Bad detection of table lines (11,32 %) (missing lines, no detection of table, detection of false table lines).
- OCR and segmentation errors (12,67 %) (e.g : 12.T instead of 12.7. Two fields are fused together which implies no given tag).
- Bad solution (1.21%) (fields are given bad tags).

The difference between global solving and local solving can be explained as the following:

In global solving, the system is processing a similar document. It has the knowledge of what it is looking for in the document. The only sources of error can be: a bad tagging (words, fields…), a bad PS extraction, a missing KWS or a missing KW and finally OCR errors.

However, in local solving, in addition to all the previous sources of errors, we can also add the bad extracted solutions. This can happen when the processed structures have no very close structures in S_DB. This can deteriorate the quality of the proposed solutions.

The results are encouraging especially if we know that a module of result verification can be added at the end of the system. This module will take every solution in every structure and check if it has a relationship with the other extracted solutions.

4/ Perspectives

Figure :

À gauche : figure d'un formulaire complet, à droite soit les XML soit les solutions sous forme graphique.

This work can be continued in several ways. First, we are going to study the "reliability" of a problem when extracted directly from a document. A "reliable" problem can help in finding a good solution. We will also focus on problem extraction from a set of homogeneous documents. Another point of interest is the database enriching. After obtaining complete solutions, the database is given new cases that have to be indexed in it. This problem of indexation is one of the most interesting problems in CBR.

5/ Conclusion

We have proposed a new approach for document analysis and interpretation. This paper shows the strength of this method in extracting data from documents, even if they have never been processed by the system before. We have applied it on administrative document (invoices, forms) and we have obtained encouraging results. This work is to be continued in order to have better results

[1]
[2]
[3]
[4]
[5]
[6]
[7]