A case-based reasoning approach for unknown class invoice processing

Hatem Hamza *,**, Yolande Belaïd**, Abdel Belaïd**
*ITESOFT, France
**LORIA, University Nancy 2. France

Abstract

This paper introduces an invoice analysis approach using Case Based Reasoning (CBR). CBR is used to analyze and interpret new invoices thanks to the previous processing experiences. Each new document is segmented into structures and interpreted thanks to a structure database. Interpreting a new document's structures relies on graph edit distance as well as on string edit distance. This paper focuses on structure extraction as well as on document interpretation via its structures interpretation. The proposed system gives an extraction and interpretation rate of 76.33%.

1. Introduction

Form and invoice analysis systems have to be fast, accurate and human independent as much as possible. The variation of information between documents makes the processing task really difficult. Two major elements can be found in invoices: tables and key-words. Table extraction and processing has been a subject of interest during the last years. Some approaches use the image pixels to find tables [5]. Some other works rely on the data extracted from the image (words, text) to detect and interpret tables. In [2], a morphological 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 intepretation: "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]. Document analysis using key-words covers many research aspects. While some works focused on the classification of documents using key-words [7], some other works[6] [3] used them to analyze and interpret the information contained in forms and invoices. The approach proposed in this paper focuses on both of these tasks. It processes different documents without any prior knowledge on them. The main idea of this paper is to analyze and interpret documents via the analysis and interpretation of each structuring elements (tables and key-words association). This paper is organized as the following: section 2 introduces briefly CBR and its use in our system. Sections 3 and 4 present our system's architecture. Finally, section 5 shows the obtained results, their interpretation and some perspectives.

2. Case-based reasoning

CBR is a powerful problem solving strategy that uses previous experiences to process new given problems [1]. A "Problem" is the input of any CBR system. It is the first component of a case in the CBR terminology (a case=problem, solution). Its resolution (to find the solution) consists in three main phases: similar case retrieval from the database, adaptation of the solution of the similar case to the studied problem and learning of new solved cases.

In our approach, we define two types of cases, which correspond to two types of structuring elements. The flow of our approach, as shown in 1, is based on two main steps: problem elaboration and local solving.

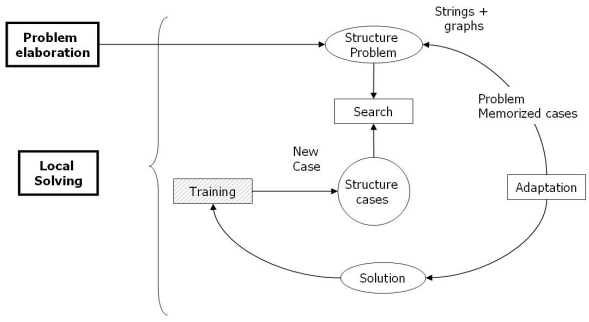

Figure 1. Flow of our approach

Problem elaboration consists in information extraction from the document. These indices are either key-words (KW) (e.g. "total", "street", "amount") and their spatial relationships, or table rows. It is obvious that the extraction of these indices without any interpretation is useless. The solution corresponds then to the interpretation of the

extracted information. In order to have an interpretation of the whole document, our system processes it structure by structure (local solving). In this paper, we focus on problem elaboration and local solving. We show that even if all the invoices belong to new classes, our system is able to extract and interpret the data contained in these documents.

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 etxraction

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 type. The attribute "type" 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 type. The type of a field is deduced from its words' types. 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. Figure 2 shows a field (small box), a horizontal line (in graytone) and a vertical block (in the bold box). 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: 'C', 'B', 'B' and 'C', then the pattern is "CBBC". 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 3 shows a document containing 4 KWS and a PS. The KWS are in gray-tone, whereas the PS is the bold box.

Figure 2. A field, a horizontal line, and a vertical block

3.2. PS extraction

PS are consecutive horizontal lines having similar patterns. This is the case of a table. Figure 3 shows a document containing a PS composed of 4 horizontal lines having the pattern "ACAAAAA". This means that there are two numerical columns, one alphabetical column and four other numerical columns.

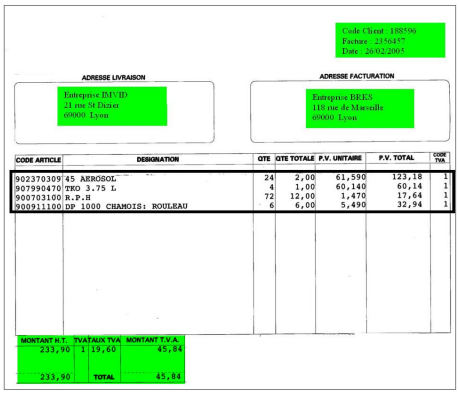

Figure 3. An invoice containing 4 KWS and a PS

The PS extraction process contains three steps:

- For each horizontal line, a list of neighbour lines 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 4, the edit distance between the patterns is null, as they represent the same string "ABB". 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 of neighbour lines HLN a new matching value is computed. This value depends on the number of exact vertical alignment of fields having the same tag. The final matching

value is the ratio:

$$RT = \frac{|matching\ fields|}{|fields\ in\ HLN|}$$

where $|X|$ is the number of elements in X. 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	B	
A		B	B

Figure 4. 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 key-words 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 key-words. For example, when the key-words ("Total", "'tax", and "Amount") are extracted together in a KWS, we know in advance that a relation exists between these words. This is different from extracting and interpreting each word separately.

3.4 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 key-word. For example, the solution of "street" is the name of the street and the number corresponding to the address (20 Albert street). In this case, KWS solution is the set of the key-words' 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;

4. Local Solving

The system builds a solution based on the structures already processed in other documents and stored in a structure database.

4.1. KWS Solving

The solving procedure acts as the following.

For each structure in the document, the nearest structure in the database is retrieved. The problem is compared to the KWS cases of the database. The solution of the nearest structure is adapted. Graph edit distance is used to find the nearest case. We used edit distance as we look for graph isomorphism, or at least, subgraph isomorphism. The cost function used to compute the graph edit distance has the same cost on vertices' edit operations and edges' edit operations. Other cost functions will be studied in the future.

The nearest structures' solutions are now adapted to the document structures. As the cases in the database have already a correct solution, the adaptation consists in taking the solution of each KW and trying to find a corresponding solution in the processed document. For example, if the solution corresponding to a KW "total" in the database 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. If a KW can not be solved, some universal knowledge related to these KW can be used. For example, it is usual that the KW "total" is followed by a numerical. The precise nature of this numerical (real, integer) depends on the document, but this information (numerical) is always valid. A rule basis containing general rules associated with key-words was built, in order to complete any partial solution of a KWS. This basis helped us completing some KWS solutions. We note here that this rule basis is not sufficient on its own to interpret a KWS. As rules are very general and are not related to any concrete case, the rule basis constitutes just a help to the system to find a solution. If no solution is found by the system, the user can then propose a solution, which will be learnt by the system (by enriching the database) in order to avoid the human intervention in other cases. The example in 5 shows a KWS which nearest KWS in the database resolves four out of five KW. By using RB, a complete solution can be found. The learning step is done by injecting each complete solution in the database. This step is still under study in order to have a more intelligent learning.

4.2. PS Solving

Each extracted PS is compared with the database cases to retrieve the nearest structure. As PS are represented with

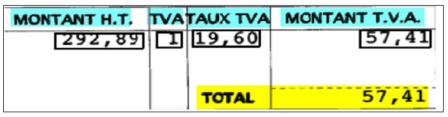

Figure 5. A KWS. Only the KW Total is solved by the rule basis.

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.

5. Experiments and future works

Our approach was tested on 800 documents. Local solving is performed on KWS and PS. The structure database is enriched gradually by the solved structures. The more structure cases are processed, the more the database becomes rich, and the more solving becomes easier for the system. The 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 described thanks to three different measures as in 1. In this equation, X can be a document, a KWS or a PS.

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

A recovered solution corresponds to a KW's solution or to a field in a PS that has been correctly extracted and interpreted. The results are given in table 1.

	R_{doc}	R_{kws}	R_{ps}
Local Solving	76.33%	76.38%	76.28%

Table 1. Results of our approach

In KWS local solving, errors are due to: 16.57% of system errors (bad solution, no solution found, confusion with other solutions) and 8.08% of OCR errors.

In PS local solving, errors are due to: 16.66% of system errors (a bad detection of table lines or a bad proposed solution, missing lines, no detection of table) and 7.14% of OCR and segmentation errors (for example, the word 23.T is read by the OCR as 23.7).

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).

The results are satisfying from an industrial point of view as we are working on invoices of unknown classes i.e no relation between a document and the document one exists. However, these results can be improved in many ways. Two immediate perspectives are being studied. We will focus on problem elaboration and especially on PS extraction. Table headers will be used in addition to the horizontal lines patterns. In this way, we can even consider PS as a special case of KWS. The learning step as well as structure database indexing are important steps in this work. After thousands of processed documents, the structure databse can contain thousands of structure cases which have to be indexed in such a way case retrieval remains always fast and accurate.

6. Conclusion

In this paper, a CBR approach for multi class invoice processing was proposed. The different processing steps, starting by document structuring, and finishing by document interpretation were exposed. Some improvements need to be done in order to enhance the results. The final use of this system will be the processing of invoices from both unknown and known classes.

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] 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.

[6] 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.

[7] 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.