Entity Extraction and Correction Based on Token Structure Model Generation

Najoua Rahal[1(✉)], Mohamed Benjlaiel[2], and Adel M. Alimi[2]

[1] Tunis el Manar University, FST, Tunis, Tunisia
najoua.rahal.tn@ieee.org
[2] Sfax University, ENIS, Sfax, Tunisia
benjlaiel@yahoo.fr, adel.alimi@ieee.org

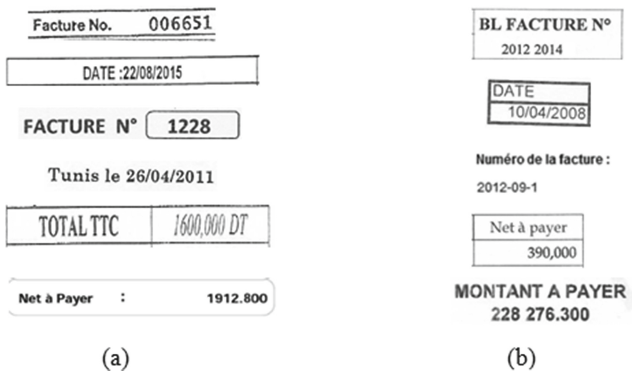
Abstract. The logical and semantic structure analysis is a basic process for invoice understanding. Be able to carry out a robust layout analysis is very difficult due to highly heterogeneous invoice templates. In this paper, we propose a local structure for entity extraction and correction from scanned invoices. It attempts to extract entity in contiguous and noncontiguous structure by automatic finding the local structure of each entity without structure model matching and user intervention. Firstly, the entities are labeled in OCRed invoice. Combining labeled entities with geometric and semantic relations, token structure models are generated. These models are used for entity extraction and mislabeling correction by ignoring some superfluous tokens detected by labeling step. The correction module to the contiguous structure differs from that of the noncontiguous structure. The obtained results with a dataset of real invoices are reported in experimental section.

Keywords: Contextual search · Contiguous and noncontiguous structure · Mislabeling correction · Token structure models

1 Introduction

In accordance with [1], Automatic document processing refers to three main categories; doctype classification, data capture/Functional Role Labeling, and document sets. Doctype classification is to assign a document image to a prestored template. Data capture represents the extraction of relevant human understandable information from document image. The category Document sets relates between documents and their contents depending on business logic. In this paper, we focus on automatic data capture from invoices regardless of their high geometric variations.

Figure 1 shows some examples of entities in contiguous (Fig. 1(a)) and noncontiguous (Fig. 1(b)) structure. It illustrates how closeness, direction and graphical elements may differ in conjunction Reference Words (RWs) e.g., “FACTURE N°”, “Date”, “Net à payer”, etc. with Key Fields (KFs) e.g., “006651”, “22/08/2015”, “228 276.300”, etc. for an entity, in various invoices.



Facture No. 006651

DATE :22/08/2015

FACTURE N° 1228

Tunis le 26/04/2011

TOTAL TTC 1600,000 DT

Net à Payer : 1912.800

(a)

BL FACTURE N°
2012 2014

DATE
10/04/2008

Numéro de la facture :
2012-09-1

Net à payer
390,000

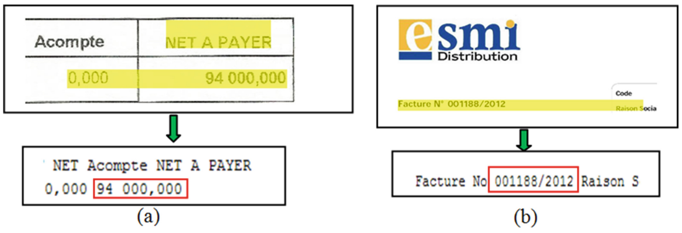
MONTANT A PAYER
228 276.300

(b)

Fig. 1. Sample of entities showed the diversity of layout styles used in invoice. (a) Entity in contiguous structure. (b) Entity in noncontiguous structure.

In this context, many initiative works, like [2], learn a local structure layout from training document and reuse it for extracting the fields in the test document. The weakness of such work is that require the human intervention for labeling semantic fields. Authors in [3,4] propose to correct the mislabeling by adding the missing labels. However, they require high regularity of structures and automatic blocks and segments obtained by OCR (Optical Character Recognition). Also, the mislabeling correction is based on matching a structure graph with a model graph.

Experimental studies have shown that the mismatch between unstructured data obtained by the OCR, like Tesseract[1] (OCR without layout analysis), and its physical representation generates another type of mislabeling called superfluous tokens. This problem is caused by mishandling of spaces by OCR. Figure 2 shows sample of entities with superfluous tokens mislabeling. At the bottom of Fig. 2(a), there is the result of labeling applied to the text of OCR to extract the "Balance Due". At the top, there is the physical representation of this labeling.



Acompte NET A PAYER
0,000 94 000,000

```
NET Acompte NET A PAYER
0,000 94 000,000
```

(a)

esmi Distribution

Facture N° 001188/2012 Code Raison Socia

```
Facture No 001188/2012 Raison S
```

(b)

Fig. 2. (a) Entity in noncontiguous structure with superfluous tokens. (b) Entity in contiguous structure with superfluous tokens (Color figure online)

[1] https://github.com/tesseract-ocr.

The final impact of this mishandling is a wrong extraction of an entity in which "0,000" represents a noisy token.

In an earlier work [5], we have proposed a method for treatment entity in contiguous structure. However, it does not able to extract entity in noncontiguous structure. Also, it does not benefit of physical and logical structure of the entity in the invoice which is the purpose of this work. The ultimate goal of our method is to extract only the relevant tokens from an entity (framed in red color) and increases the accuracy of the extraction process.

Our contributions are: (i) a robust system for entity extraction based on contextual search of local structure of each entity. Then, there is no need to classify contiguous and noncontiguous structure. Our system starts its contextual search in contiguous structure. If no result is found, then, it moves automatically to the treatment of extraction entity in noncontiguous structure. (ii) Adoption of correction step to eliminate the superfluous tokens caused by the labeling step.

In the remainder of our paper, we firstly describe in detail our solution. Next, we discuss obtained experimental results. Finally, the paper is concluded.

2 Proposed Method

The overview of our proposed method is given by Fig. 3. Firstly, invoice image is processed by OCR engine. Secondly, entities are labeled in OCRed invoice image. Once labeled, the local structure of each entity is detected. For each entity structure, a token model is generated which aims to eliminate the superfluous tokens caused by the labeling step. In this model, the KFs represent the tokens and the distances represent the relationships between them. Finally, an incremental algorithm is applied to concatenate each two consecutive tokens of which the distance between them respects a certain threshold.

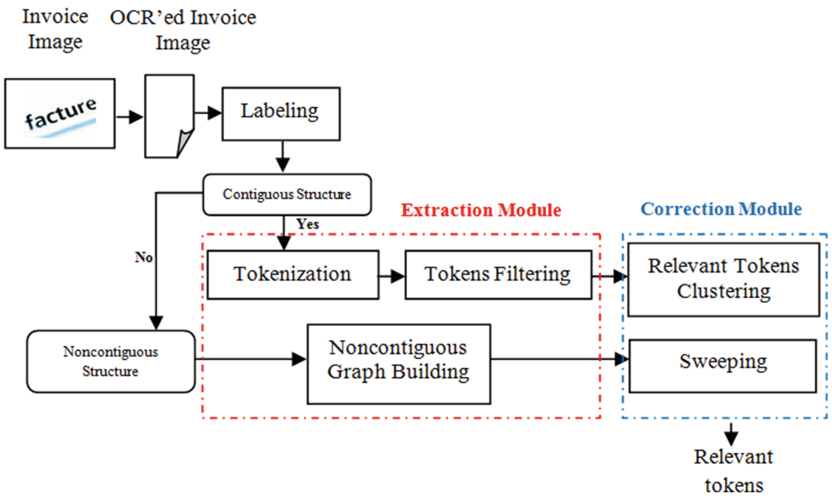


Fig. 3. Global schema of proposed method

2.1 Labeling

Entities are labeled in the invoice using Patterns of Regular Expressions (Regex). Each invoice I is defined as:

$$I = \{L_i\} \tag{1}$$

Where $\{L_i\}$ is a set of labels. Each label is represented by:

$$L_i = \{R_i, F_i\} \tag{2}$$

Where R_i is the Reference Words (RWs) label and F_i is the Key Fields (KFs) label.

2.2 Extraction Entity in Contiguous Structure

Tokenization. The tokenization allows the presentation of a label in the form of a set of tokens, $SetT$, which are separated by whitespace character as:

$$L_i = SetT \tag{3}$$

The tokens of R_i are defined as:

$$SetR = \{T_i^R | SetR \in SetT\} \tag{4}$$

The tokens of F_i are defined as:

$$SetF = \{T_i^F | SetF \in SetT\} \tag{5}$$

Tokens Filtering. In this step, Algorithm 1 is iteratively used to delete $SetR$. This algorithm is stopped when $SetR$ is empty. At this stage, $SetT$ contains only the entire tokens $SetF$. Each token is stored on its bounding box which is defined by:

$$T_i^F \rightarrow [x_i^F, y_i^F, w_i^F, h_i^F] \tag{6}$$

Where x_i^F and y_i^F represent the coordinate of upper left corner, w_i^F the width and h_i^F is the height of the rectangle.

Algorithm 1. Tokens filtering

```
1: **Input:** $SetT = SetR.SetF$ // Set of tokens
2: **Output:** $SetT = SetF$
3: **begin**
4:     **while** $SetR \neq \emptyset$ **do**
5:         $SetT = SetT/T_i^R$
6:     **end while**
7:     return $SetT = SetF$
8: **end**
```

Relevant Tokens Clustering. In this step, we propose a correction module for elimination the superfluous tokens. It represents the arrangement of relevant tokens of local entity. The geometric relations of a structure is modeled by distances measuring. The clustering of relevant tokens and eliminating the noisy content require the distances measuring between consecutive tokens T_i^F and T_j^F $(j = i + 1)$. Each distance is calculated as:

$$d_{ij} = x_j^F - (x_i^F + w_i^F) \tag{7}$$

The incremental algorithm, detailed in Algorithm 2, is applied to concatenate relevant tokens. $SetF$ contains at least one token. In this case, the latter represents the relevant token. If $SetF > 1$, then, we need to cluster relevant tokens. To achieve this goal, a threshold S is defined as the maximum distance between two consecutive tokens. This threshold is empirically defined. The measured distance is compared with S. If $d_{ij} \leq S$, then, the tokens are concatenated. If it is not the case, then, the algorithm is stopped and the rest of tokens, $SetN$, are ignored.

Algorithm 2. Incremental algorithm

```
1: Input: $SetF$ // Entity containing at least one token and may contain noisy
tokens $SetN = \{T_i^N | SetN \subset SetF\}$
2: Output: $RT = SetF/SetN$// Relevant tokens
3: begin
4:    while $SetF \neq \emptyset$ do
5:      if $SetF = 1$ then
6:        $RT = SetF$
7:      elseif $SetF > 1$ then
8:          for $j = 1 : SetF - 1$ do
9:            if $d_{ij} \leq S$ then
10:             $T_i^F = concat(T_i^F, T_j^F)$
11:             $T_j^F = []$
12:           elseif $d_{ij} > S$ then
13:               $RT = SetF/SetN$
14:           end if
15:         end for
16:         return $RT = SetF/SetN$
17:     end if
18:   end while
19: end
```

2.3 Extraction Entity in Noncontiguous Structure

Entity in noncontiguous structure means that RWs and KFs appear in the invoice in vertical structure. Since the drawing of relationships between RWs and all the KFs is time consuming and no avail, we propose to filter the labels. This requires the detection of KFs in a given region.

For relevant entity extraction, we build a graph of structural relationships. This graph is called Noncontiguous Graph.

Noncontiguous Graph Building. For noncontiguous entity structure extraction, as detailed in Algorithm 3, a graph is built $G = (N, M, E)$ in which N is a node of the label R_i. M is a set of finite nodes that represent the labels F_j having the centers under the center of N. $E \subseteq N \times M$ is a finite set of arcs which represent a geometric relationships between the node N and the nodes of M. Each arc $e_{ij} \in E$ relating the node N and m_j is represented by Nm_j. We define a feature vector which describes the geometric relationships between N and m_j.

$$a_{ij} = (CN_i, Cm_j, e_{ij}) \tag{8}$$

Where: CN_i is the center of the node N (step 4 in Algorithm 3). Cm_j is the center of each node m_j (step 6 in Algorithm 3). e_{ij} is the distance that separates the bounding boxes of the labels corresponding to N and m_j (step 10 in Algorithm 3), as we can view in Fig. 4. The idea is to detect the nearest m_j to N (step 13 in Algorithm 3). We consider only the nodes having the centers under the center of N. The distances are calculated as:

$$e_{ij} = \begin{cases} 1, & \text{if } Cm_j(2) > CN_i(2) \\ 0, & \text{else} \end{cases} \tag{9}$$

Where: $Cm_j(2)$ is the second coordinate (ordinate) of the center Cm_j. $CN_i(2)$ is the second coordinate of the center CN_i. e_{ij} is calculated to filter the KFs labels i.e., we bethink only the centers having the upright under the center of N.

The centers are calculated to determinate the nearest m_j to N. In Fig. 4, m_4 represents the nearest label KFs node to N i.e., m_4 is the relevant KFs. However, the latter may contain noisy tokens that must be eliminated. So, we need to tokenize the relevant KFs (step 14 in Algorithm 3) for clustering relevant tokens and ignore the noisy one.

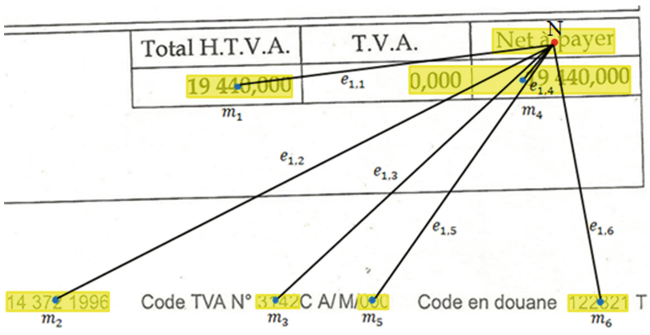


Fig. 4. Noncontiguous graph

The difficulty of detecting the relevant tokens of field in a vertical structure resides in this step. In a horizontal structure, the starting token from which begins the clustering of tokens is known. In addition, noisy tokens are found only on the right side. By cons, in a vertical structure, it is first necessary to determine the starting token. Then, we have to perform a sweeping to eliminate noisy tokens to the left and then to the right. To achieve this goal, a subgraph of relationships is built between the node N and the tokens $K = \{k_j\}$ of the nearest node m_4. The nearest token is the frame used for a sweeping. To determinate the nearest token, we calculate the distance p_{ij} between the node N and each token k_j (step 16 in Algorithm 3). The nearest token possesses the minimum distance with N (step 18 in Algorithm 3). We call this token "ind" as it represents an index from which begins the sweeping. In Fig. 5, the "ind" is k_2.

Sweeping. The sweeping is the exploration token by token of an entity. It is done in both directions to the left (step 20 in Algorithm 3) and then to the right (step 23 in Algorithm 3) for superfluous tokens elimination. The geometrical relationships, provided by the distances measured between tokens inside $Left\_M$, are used to concatenate relevant tokens. This matrix is defined as:

$$Left\_M = (K(1 : ind)) \tag{10}$$

Whenever, we calculate two distances between two consecutive tokens. The first distance is calculated as:

$$n_{Z-1,Z} = Left\_M_{(Z,1)} - (Left\_M_{(Z-1,1)} + Left\_M_{(Z-1,3)}) \tag{11}$$

This distance must not exceed the threshold S previously identified (explained in Sect. 2.3).

To ensure the horizontal alignment of consecutive tokens, we need to calculate the distance between their second coordinates. This distance must not exceed certain threshold H and is calculated as:

$$g_{Z-1,Z} = Left\_M_{(Z,2)} - Left\_M_{(Z-1,2)} \tag{12}$$

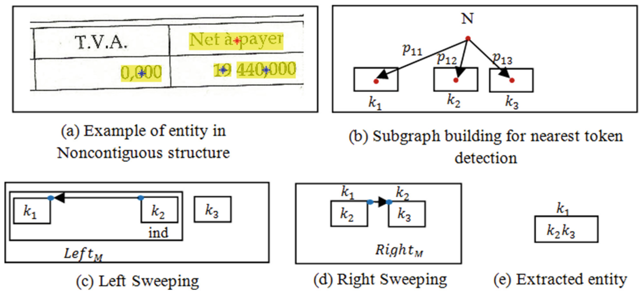


Fig. 5. Subgraph of tokens

The left sweeping outcome is $Left\_M$ containing only one element grouping the relevant tokens. This element is added to the beginning of the created matrix $Right\_M$ for the right sweeping. So, all relevant tokens are grouped in $Right\_M$ (step 25 in Algorithm 3) which is defined as:

$$Right\_M = (K(Left\_M + 1 : end)) \tag{13}$$

The concatenation in the right sweeping is done with the same principles detailed in the left sweeping. Figure 5 shows the process of sweeping for tokens concatenation. In Fig. 5(b), a subgraph of geometric relationships is established between the nodes. The nearest token, ind, having the minimum distance with the node N is detected. The latter is "19". In Fig. 5(c), $Left\_M$ contains two tokens "0,000" and "19". The distance $n_{Z-1,Z}$ between these tokens exceeds the threshold S. For that, the token "0,000" is eliminated. In Fig. 5(d), $Left\_M$, containing one element, is integrated in the start of $Right\_M$ and the right sweeping begins. In Fig. 5(e), the right sweeping allows the concatenating of relevant tokens ("19", "440,000").

3 Experiments

3.1 Dataset

For test, we use a dataset of 930 real invoices obtained from Compagnie des Phosphates de Gafsa (CPG)[2]. The entities are categorized into 7 types: Invoice Number (N°), Invoice Date (DT), Account Identity (AI), Pre-tax Amount (PA), Total Including Tax (IT), Holdback (H) and Balance Due (BA).

Algorithm 3. Noncontiguous entity structure extraction

```
1: Input: $N$ // RWs
$M$ // KFs
$K$ // Tokens
2: Output: $RT$// Relevant tokens
3: begin
4:     $CN_i = calculate\_Center(N)$
5:         for all $m_j$ do
6:             $Cm_j = calculate\_Center(m_j)$
7:         end for
8:         for all $m_j$ do
9:             if $Cm_j(2) > CN_i(2)$
10:                $e_{ij} = Euclediandist(CN_i, Cm_j)$
11:            end if
12:        end for
13:        $RF = m_j(dmin(e_{ij}))$
14:        $K \leftarrow RF$
```

[2] http://www.cpg.com.tn.

```
15:        for all $k_j$ do
16:            $p_{ij} = Eucledian dist(CN_i, Ck_j)$
17:        end for
18:        $ind = k_j(dmin(p_{ij}))$
19:        while $Left\_M \neq \emptyset$ do
20:            left_sweep()
21:        end while
22:        while $Right\_M \neq \emptyset$ do
23:            right_sweep()
24:        end while
25:        $Right\_M = RT$
26: end
```

It is important to indicate that our system sustains the data extraction from grayscale, color and bi-tonal (black and white) images. Our system is insensitive to the multiplicity of fonts. Although the preprocessing step does not belong of our work, our system is able to manipulate little noisy invoices with a slight skew. These invoices contain graphical elements, logos, vertical and horizontal lines, and tables. Some manipulated invoices are shown in Fig. 6.

We have used our ground truth to evaluate our system's performance. This ground truth was manually prepared.

3.2 Erroneous RWs Correction

In our system, entities are labeled using Regex. The patterns are written to allow some OCR errors in RWs such as confusing zero with capital or lowercase O (e.g., Facture no$\Rightarrow$ Facture n0). This can allow unconstrained input that nearly matches the Regex pattern to be taken in account and significantly improve the performance. The refined Regex has allowed us to detect correctly 62 N$^{\circ}$, 13 DT, 53 PA, 17 IT, 153 AI, and 23 BA.

3.3 Structure Correction Evaluation

To capture contiguous and noncontiguous structures of entity, a set of Regex patterns are used in conjunction with geometric relationships between labels. The correction step is integrated for superfluous tokens elimination. The goal is to increase the accuracy of the extraction. The correction step has allowed us to correct to 100 % the superfluous tokens and yielded a growth of the accuracies. Table 1 shows the impact of this step for each entity. For that, we use two options; without correction (W/o C) and with correction (With C). The most interesting result is for the N$^{\circ}$ entity since it has a countless number of formats. The obtained rates justify the fixed threshold distances between any consecutive tokens. We use two thresholds: S is around 32. It is fixed for concatenating the tokens whatever in contiguous or noncontiguous structure. The second threshold H is around 12. It is fixed only in noncontiguous structure to ensure the alignment and the consecutiveness of the tokens. The thresholds set show the power possessed by

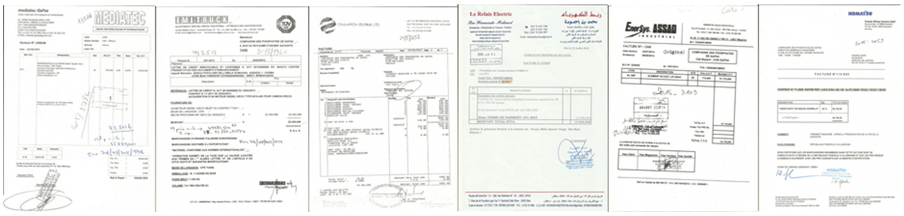


Fig. 6. Sample of invoices in our dataset

our method for correction. To ensure the robustness of our correction method, we propose to strengthen these thresholds by other features such as font size to avoid bad detection that can be generated by using the few thresholds in other models.

Missed entities, as detailed in Table 2, are due to the following issues: errors in RW represent the RWs completely wrong, so, they cannot be identified by the Regex. Errors in KF are the KFs partially corrected or completely not corrected: if one field is not properly extracted, then, the entity was regarded as erroneous. The confusing labels (CL) means that the label is not associated with the correct entity which leads a failed match for another entity. The OCR sometimes missed the text zone (MT) due to: skewed image, noise, degraded characters, bad detection of tabular structure, etc.

Table 1. Impact of correction

Entity	Method (%)	
	W/o C	With C
N°	83.22	98.17
DT	97.63	98.82
BA	95.01	97.04
PA	94.01	96.72
IT	93.42	96.24
H	90.91	93.94

Table 2. Missed entities

Entity	Error Types (%)			
	RW	KF	CL	MT
N°	29.41	29.41	11.76	29.41
DT	9.10	36.36	63.64	9.10
BA	11.76	23.53	29.41	35.29
PA	41.37	20.69	6.90	31.04
IT	41.67	33.33	8.33	16.67
AI	40.48	19.05	0	40.48

Table 3. Rates comparison

	Recall(%)	Precision(%)
Method [3]	88.88	95.23
Method + Module correction [3]	93.37	97.50
Our method	92.49	98.86
Our method + Module correction	96.78	98.91

3.4 Comparison with Existing System

Table 3 synthesizes the obtained Recall and Precision of our system. In this table, we also compare our work with the results obtained by the system proposed in [3]. Recall and Precision are defined as:

$$Recall = \frac{relevant\ extracted\ entities}{relevant\ entities} \tag{14}$$

$$Precision = \frac{relevant\ extracted\ entities}{extracted\ entities} \tag{15}$$

4 Conclusion

We have proposed an approach for entity extraction from scanned invoices. We have showed how adopting a local structure of entities is very efficient for data extraction. This represents a powerful tool in dealing with variant layout entities. Our method is reinforced by correction step for superfluous tokens elimination. The experimental results have showed an interesting improvement in the performance and accuracy of the extraction process.

Acknowledgment. We are grateful to CPG Company for providing real invoices for test.

References

1. Saund, E.: Scientific challenges underlying production document processing. In: Document Recognition and Retrieval XVIII, DRR (2011)
2. Rusinol, M., Benkhelfallah, T., Poulain, V.D.: Field extraction from administrative documents by incremental structural templates. In: International Conference on Document Analysis and Recognition, ICDAR (2013)
3. Kooli, N., Belaid, A.: Semantic label and structure model based approach for entity recognition in database context. In: International Conference on Document Analysis and Recognition, ICDAR (2015)
4. Dejean, H.: Extracting structured data from unstructured document with incomplete resources. In: International Conference on Document Analysis and Recognition, ICDAR (2015)
5. Rahal, N., Benjlaiel, M., Alimi, Adel. M.: Incremental structural model for extracting relevant tokens of entity. In: IEEE International Conference on Systems, Man, and Cybernetics (SMC) (2016, to be published)