# A Domain Knowledge-based Approach for Automatic Correction of Printed Invoices

Enrico Sorio, Alberto Bartoli, Giorgio Davanzo and Eric Medvet
DI[3] - Industrial and Information Engineering Dept.
University of Trieste
Via Valerio 10, Trieste, Italy
email: bartoli.alberto@units.it

***Abstract*—Although OCR technology is now commonplace, character recognition errors are still a problem, in particular, in automated systems for information extraction from printed documents. This paper proposes a method for the automatic detection and correction of OCR errors in an information extraction system. Our algorithm uses domain-knowledge about possible misrecognition of characters to propose corrections; then it exploits knowledge about the type of the extracted information to perform syntactic and semantic checks in order to validate the proposed corrections. We assess our proposal on a real-world, highly challenging dataset composed of nearly 800 values extracted from approximately 100 commercial invoices and we obtained very good results.**

***Index Terms*—document understanding, optical character recognition, error detection, error correction**

## I. Introduction

Information extraction from printed documents plays a key role in many areas: office automation, knowledge management, intelligence, and so on. The transmission of information between different organizations is an important part of many interorganizational workflows. In many pratical cases the means of information transfer are printed documents from which human operators have to extract the desired information; this information is then inserted in another document or application. As a matter of fact, despite the widespread use of ICT technology, the interaction pattern involving printed documents and manual data entry is still very common. This is true, in particular, for short-lived interactions and interactions between organizations that cannot, or are not willing to, afford the investment required by a fully electronic workflow.

The automation of information extraction is a complex task that can be divided in two main sub-problems: the extraction of the desired information and the automatic correction of OCR errors. Regarding the former several approaches have been proposed for extracting automatically a specified set of information items from a document, once the layout is known [3], [4], [8]. The importance of the latter for the overall quality of information extraction systems is demonstrated in [11] and [9], which quantify the negative impact of OCR errors. The automatic correction of OCR errors is widely discussed in literature [7], [6], [12], [2], but is still an open research topic, expecially regarding the correction of text obtained from low quality or noisy printed documents.

We propose a solution aimed at improving the accuracy of OCR-produced results on real-world printed documents. Our solution is tailored to environments in which the type of the information to be extracted is known, as in [3], [4], [8], [10], [1]. We aim at detecting and correcting OCR errors in the extracted information using knowledge about the type of each information (date, currency and so on).

Traditionally, there are two main ways to detect OCR errors: dictionary lookup and character n-gram-matching [7], [6]. Other methods for automatic OCR errors correction, like the one described in [12] use topic models, which automatically detect and represent an article semantic context. In this work we propose a different approach based on the use of syntactic and semantic knowledge about the handled text that is often available, expecially in the information extraction systems of our interest.

Our approach is composed of two main steps: in the first step we exploit domain-knowledge about possible OCR mistakes to generate a set of variants of the string extracted by the OCR. In the second step we perform a suite of syntactic and semantic checks to select from the correct string from the set of proposed ones. We also perform an aggregate semantic check in order to verify the correctness of two or more elements that are semantically linked to each other (e.g., total, taxable amount and vat in an invoice document). This approach is able to detect and correct OCR errors also in noisy documents without introducing errors.

As far as we know there is only one approach similar to the one here proposed, but tailored to a different application domain: [5] proposes an approach where a tree grammar, employed to define syntactically acceptable mathematical formulae, is used to detect and correct misrecognized mathematical formulae.

## II. Our approach

### A. Overview

Documents of our interest are typically electronic images of paper documents obtained with a scanner. A document is associated with a schema, as follows. The schema describes the information to be extracted from the document and consists of a set of typed elements $e$: for each element, the document contains zero or one value $v$.

978-1-908320-05/6/$25.00©2012 IEEE

For example, a schema could be `date`, `totalAmount`, `documentNumber`, respectively with types `date`, `currency` and `number`; a document with this schema could contain the values `"7/2/2011"`, `"23,79"` and no value for the respective elements.

Executing an OCR procedure on the electronic image of a document we obtain a set of strings $\{l_1, l_2, \ldots, l_n\}$. For each element $e$ of the schema, we associate the candidate string $l$ to that element. The description of the system that automatically associate each element $e$ with the candidate string $l$ is beyond the scope of this paper.

For each searched element, it may be $l \neq v$, because of the following reasons:

- $l$ may contain $v$ and **extra-text** that is not part of $v$. For example $l =$ `"date:21/12/2008"` while $v =$ `"21/12/2008"`.
- $l$ may not contain $v$ due to OCR errors. These errors can be of two types:
  - *segmentation error*: different line, word or character spacings lead to misrecognitions of white-spaces, causing segmentation errors (e.g., $l =$ `"076 4352 056 C"` while $v =$ `"0764352056C"`).
  - *misrecognition of characters*: low print quality, dirt and font variations prevent an accurate recognition of characters (e.g., $l =$ `"|9,5SG"` while $v =$ `"19,556"` or $l =$ `"IOAS/0B127"` while $v =$ `"105/08127"` ).

While the segmentation and misrecognition problem may occur only with digitized documents, the extra-text problem may occur also with digitally born documents.

We propose a solution that uses a suite of syntactic and semantic checks in order to detect and correct these OCR-generated errors.

Our system is designed to be *modular* and *extensible*, so as to make it possible to augment and improve the domain-knowledge encoded in the module as well as to accommodate further application-specific rules beyond those currently embedded in the system. A high-level description of this step follows, full details are provided in the next sections.

For each element, we generate a set of values $\{v_1^*, v_2^*, \ldots, v_n^*\}$ that, due to OCR errors, might have lead to the extraction of $l$. This set is generated by applying to the extracted string $l$ a number of possible substitutions as encoded in a predefined table of *substitution rules*. This table encodes a domain-knowledge about possible misrecognitions of characters. Then, we perform a suite of syntactic and semantic checks to exclude from the previous set all those values that do not satisfy at least one of these checks. We denote by $V^*$ the resulting set of candidate values. These syntactic and semantic checks are encoded in boolean functions tailored to the type of the element to be extracted. We have implemented these functions for the following elements: `data`, `number`, `vatNumber`, `currency`, `fiscal code` (a unique id assigned to each person that lives in Italy, whose structure follows certain costraints [13]).

In addition to checks applied to individual elements, a check is performed also on groups of *correlated elements* whose values have to satisfy one or more joint conditions. For example, in our invoice scenario the group $\{\texttt{taxableAmount}, \texttt{vat}, \texttt{total}\}$ has to satisfy the condition $v_{\texttt{taxableAmount}} + v_{\texttt{vat}} = v_{\texttt{total}}$. Based on this observation, we introduced a semantic check on the coherency of the values for the elements in the group, as well as the ability to suggest a set $\mathcal{V}$ of further corrections in case the check is violated.

Finally, for each element we select a single value $v_e^*$ that will be the proposed correction, as follows: if $\mathcal{V}$ contains one or more values for $e$ the first one is selected, otherwise the first one contained in $V_e^*$ is selected. As will be explained in the next sections, if $\mathcal{V}$ or $V_e^*$ contain more than one value all have the same chance of being correct; the first one is chosen and the system notify the operator about the remaining candidate values.

Our system is provides as output a qualitative evaluation of the proposed corrections, which is a sort of confidence level for the value associated with an element. There are three quality levels, low, medium and high, that are chosen based on the cardinality of $V_e^*$ and of $\mathcal{V}$. We omit the details of the mapping for brevity. Using the quality for an element, we can decide whether to use that (possibly corrected) value automatically, or ask a confirmation to the operator. The operator is always involved when two or more equally likely corrections are selected. If the candidate string $l$, obtained from the initial OCR procedure, satisfies all checks, then no correction is attempted and an high quality level is given as output.

### B. Single element

The workflow applied to each single element is shown in Figure 1 and discussed below. Algorithm 1 shows the corresponding pseudocode. As shown in Figure1, our approach is based on components that may be extended as appropriate in a different application domain: the table of substitution rules and the functions $f^e_{\text{syntax}}(v)$ and $f^e_{\text{semantic}}(v)$ that implement syntactic and semantic checks on an element value $v$ and are tailored to each type of element $e$. The table of *substitution rules* $S$ is a set of pairs $s \rightarrow \mathcal{S}$, where $s$ is a string and $\mathcal{S}$ is a set of strings. Each element of $\mathcal{S}$, say $s'$, is a string visually similar to $s$, i.e., such that the OCR might have extracted $s$ instead of the correct value $s'$. An example of substitution rule used in our system is `"1"` $\rightarrow$ $\{$`"i"`, `"I"`$\}$, which means that the OCR might extract the character `"1"` while in fact the correct character might be either `"i"` or `"I"`. Another example is `"11"` $\rightarrow \{$`"N"`$\}$, which means that the extracted string might be `"11"` while the correct value might be `"N"`

In detail, the processing of each element is as follows.

1) The *string cleaning* stage removes from the input string $l$ the characters that satisfy both the following conditions: the character cannot be present in $v$ (according to $f_{\text{syntax}}$) and is not present in any left-side element of $S$.
2) The *substitution application* stage generates a set of candidate values $V'$ applying all the substitution rules

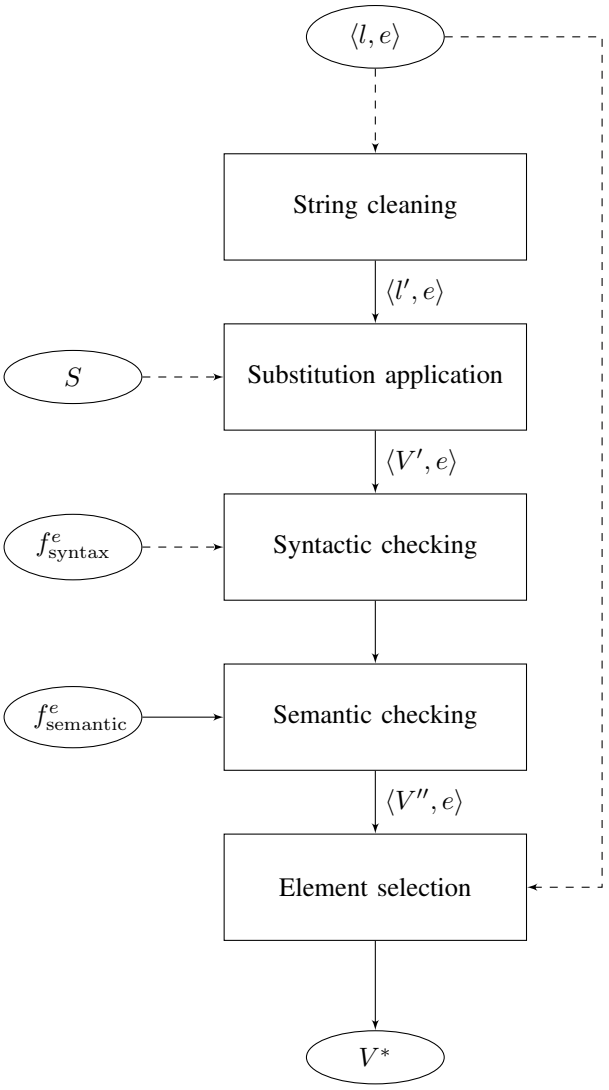



Figure 1. OCR error fixer workflow.

on the input string $l'$. For example, with $l' =$ `"a1b1"` the substitutions can be:

- *One-to-one*: for example, the rule `"1"` $\rightarrow$ `"i"` provokes the insertion in $V'$ of the candidate values:

$$\{\texttt{"a1b1"}, \texttt{"aib1"}, \texttt{"a1bi"}, \texttt{"aibi"}\}$$

- *One-to-many*: for example, the rule `"1"` $\rightarrow$ $\{$`"i"`, `"I"`$\}$ provokes the insertion in $V'$ of the candidate values:

$$\{\texttt{"a1b1"}, \texttt{"a1bi"}, \texttt{"a1bI"}, \texttt{"aib1"}, \texttt{"aibi"}, \texttt{"aibI"}, \texttt{"aIb1"}, \texttt{"aIbi"}, \texttt{"aIbI"}\}$$

3) The *syntactic* and *semantic* checking stage removes from the set of candidate values $V'$ those strings that do not satisfy one or both of the semantic checks for the element (joint checks applied to groups of correlated elements are discussed in the next section) and generates $V''$.

4) The *element selection* stage selects the strings with minimal distance from the extracted string $l'$. Since the Levenshtein distance gives the same weight to a character replacement and a character removal, we used a modified version of that distance ($d_{\text{L}}$) which give more weight to a character removal. This is done adding the length difference between strings to the original Levensthein distance.

If the final set of candidate values $V^*$ is composed of more than one element, then the operator is asked to make the final choice. Otherwise, the operator may or may not be involved depending on the quality level assigned to the element, as outlined in the previous section.

**for** $e$ in $\{e_1, e_2, \ldots, e_n\}$ **do**
  $l_e$ := extracted string for $e$
  $l'_e := f_{\text{cleaner}}(l_e)$
  $V'_e := f_{\text{substitutions}}(l'_e)$
  $V''_e := \emptyset$
  **for** $v'$ in $V'_e$ **do**
    **if** $f^e_{\text{syntactic}}(v')$ **and** $f^e_{\text{semantic}}(v')$ **then**
      $V''_e := V''_e \cup v'$
    **end if**
  **end for**
  $V^*_e := \emptyset$
  $d_{\min} := \min_{v'' \in V''_e} d_{\text{L}}(v'', l'_e)$
  **for** $v''$ in $V''_e$ **do**
    **if** $d_{\text{L}}(v'', l'_e) = d_{\min}$ **then**
      add $v''$ to $V^*_e$
    **end if**
  **end for**
**end for**

**Algorithm 1:** Single element fixing pseudo-code

### C. Correlated element fixing

Given a group $E = \{e_{i_1}, e_{i_2}, \ldots, e_{i_k}\}$ of correlated elements, we introduce a joint semantic check encoded in a boolean function

$$f^{e_{i_1}, e_{i_2}, \ldots, e_{i_k}}_{\text{semantic}}(v^*_{i_1}, v^*_{i_2}, \ldots, v^*_{i_k}) \Rightarrow \{T, F\}$$

The joint conditions are used to provide further corrections: e.g., $v_{\texttt{vat}} = v_{\texttt{total}} - v_{\texttt{taxableAmount}}$.

Using the single element fixing algorithm we obtain, for each element $e_i$, a set of candidate values $V^*_{e_i}$. Using the sets $V^*_{e_i}$ we generate all the possible combination of values $\langle v^*_{e_{i_1}}, \ldots, v^*_{e_{i_k}}\rangle \in V^*_{e_{i_1}} \times \cdots \times V^*_{e_{i_k}}$.

Then we create a new set of tuple called $\mathcal{V}$ where we store all the combinations $\langle v^*_{e_{i_1}}, \ldots, v^*_{e_{i_k}}\rangle$ that satisfy the joint semantic check.

We also define $k$ functions $\mathcal{F}_j(v^*_{e_{i_1}}, \ldots, v^*_{e_{i_{j-1}}}, v^*_{e_{i_{j+1}}}, \ldots, v^*_{e_{i_n}}) = \hat{v}_{e_{i_j}}$, i.e., having $k-1$ elements, the function $\mathcal{F}_j$ gives the $j$-th value $\hat{v}_{e_{i_j}}$ such that $f_{\text{semantic}}$ is satisfied.

Using these functions, we can obtain a value $\hat{v}_{e_{i_j}}$ for at most one element $e_{i_j}$ for which $V^*_{e_{i_j}} = \emptyset$.

In particular, in our scenario we have a single

$$E = \{\texttt{total}, \texttt{taxableAmount}, \texttt{vat}, \texttt{rate}\}$$

for which the joint semantic function is defined as follows (we omit the elements superscripts for ease of reading):

$$f(v^*_{\texttt{total}}, v^*_{\texttt{taxableAmount}}, v^*_{\texttt{vat}}, v^*_{\texttt{rate}}) = \\ = \begin{cases} T, & \text{if } v^*_{\texttt{total}} = v^*_{\texttt{taxableAmount}} + \\ & v^*_{\texttt{vat}} \wedge v^*_{\texttt{vat}} = \frac{v^*_{\texttt{taxableAmount}} * v^*_{\texttt{rate}})}{100} \\ F, & \text{otherwise} \end{cases}$$

Moreover, for the $E$ described above we can define these functions:

$$\mathcal{F}_{\texttt{total}} = v^*_{\texttt{taxableAmount}} + v^*_{\texttt{vat}}$$
$$\mathcal{F}_{\texttt{taxableAmount}} = v^*_{\texttt{total}} - v^*_{\texttt{vat}}$$
$$\mathcal{F}_{\texttt{vat}} = v^*_{\texttt{total}} - v^*_{\texttt{taxableAmount}}$$
$$\mathcal{F}_{\texttt{rate}} = \frac{100 \cdot v^*_{\texttt{vat}}}{v^*_{\texttt{taxableAmount}}}$$

**for each** $\{e_{i_1}, e_{i_2}, \ldots, e_{i_k}\}$ **do**
  **for** $\langle v^*_{e_{i_1}}, \ldots, v^*_{e_{i_k}} \rangle$ in $V^*_{e_{i_1}} \times \cdots \times V^*_{e_{i_k}}$ **do**
    **if** $\exists j \mid V^*_{e_{i_j}} = \emptyset$ **then**
      $\hat{v}_{e_{i_j}} = \mathcal{F}_j(v^*_{e_{i_1}}, \ldots, v^*_{e_{i_j-1}}, v^*_{e_{i_j+1}}, \ldots, v^*_{e_{i_k}})$
      add $\langle v^*_{e_{i_1}}, \ldots, \hat{v}_{e_{i_j}}, \ldots, v^*_{e_{i_k}} \rangle$ to $\mathcal{V}$
    **else**
      **if** $f^{e_{i_1}, e_{i_2}, \ldots, e_{i_n}}_{semantic}(v^*_{e_{i_1}}, v^*_{e_{i_2}}, \ldots, v^*_{e_{i_k}}) = T$ **then**
        add $\langle v^*_{e_{i_1}}, \ldots, v^*_{e_{i_k}} \rangle$ to $\mathcal{V}$
      **end if**
    **end if**
  **end for**
**end for**

**Algorithm 2:** Correlated element fixing algorithm

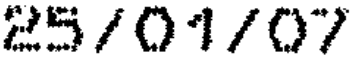
## III. Experiments and results

### A. Dataset

In order to assess our approach effectiveness we collected two real-world datasets: the first ($D_1$) composed by 591 values (extracted from 76 invoice documents) and the second ($D_2$) composed by 208 values (extracted from 37 invoice documents).

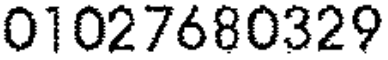
All documents belonging to the dataset were digitalized after being handled by a corporate environment, thus they are quite noisy with handwritten signatures, stamps, etc.; during the digitalization the pages were positioned in a variable way with respect to the scanner area, resulting in images whose content position is variable.

We defined a schema composed of 9 elements: `date`, `invoiceNumber`, `total`, `taxableAmount`, `vat`, `rate`, `customer`, `customerVatNumber`, `issuerVatNumber`.



OCR → "R5/0 I/07"
$v$ = "25/01/07"

(a)



OCR → "O1027eSO329"
$v$ = "01027680329"

(b)

OCR → "QXOVANNX00"
$v$ = "GIOVANNI**"

(c)

Figure 2. Examples of OCR errors; in particular, Figure 2(a) is a typical low quality dot matrix print.

The ground truth for the entire dataset was constructed using the defined schema. Each document image is converted to a set of values using an OCR system[1]. Then an operator inspected visually each document and, for each element of the schema, manually selected the corresponding string (i.e., $l$), if present, and wrote down the correct value (i.e., $v$).

The dataset $D_2$ was created with the purpose of testing our algorithm with low quality and dot-matrix printed documents. OCR performances on dot-matrix printed document is poor: a visual inspection of the dataset showed us that the OCR system introduced a sensible amount of errors, making the scenario highly challenging. In particular, approximately 55% of the strings recognized by the OCR system were wrong. Figure 2 shows some portions of documents that led to OCR errors.

### B. Performance evaluation

The ground truth for this experiment is hence composed, for each document, of the following set of pairs:

$$L = \{\langle l_{e_1}, v_{e_1} \rangle, \langle l_{e_2}, v_{e_2} \rangle, \ldots, \langle l_{e_n}, v_{e_n} \rangle\}$$

Each pair associates the string $l$ with the correct value $v$ wrote down by the operator.

We applied our algorithm to each set $L$ and counted the number of times when the output $v^*$ was equal to $v$, i.e., the output was correct.

Table I and II shows the performance of our algorithm in both datasets. In particular, the tables show the percentage of

[1]CuneiForm 1.1.0, an open source document analysis and OCR system.

| Element | Correct before | Correct after | Total |
|---|---|---|---|
| date | 46.05% | 89.47% | 76 |
| invoice number | 28.95% | 55.26% | 76 |
| vat number | 40.00% | 88.89% | 135 |
| taxable amount | 47.37% | 90.79% | 76 |
| total | 30.26% | 85.53% | 76 |
| vat | 57.89% | 96.05% | 76 |
| rate | 100.0% | 100.0% | 76 |

Table I
PERFORMANCE ON DATASET $D_1$

| Element | Correct before | Correct after | Total |
|---|---|---|---|
| date | 54.8% | 77.4% | 31 |
| invoice number | 46.1% | 50.0% | 26 |
| vat number | 19.12% | 58.82% | 68 |
| taxable amount | 46.4% | 71.4% | 28 |
| total | 43.4% | 69.6% | 23 |
| vat | 29.1% | 45.8% | 24 |
| rate | 50.0% | 75.0% | 8 |

Table II
PERFORMANCE ON DATASET $D_2$

correctly recognized values before (second column) and after (third column) the execution of our procedure.

Averaging these values we obtain the global percentage of correct values before and after the application of our algorithm. In dataset $D_1$ we increment correctness from 49.1% to 86.8%, while in dataset $D_2$ form 31.7% to 61.23%. In summary, our system is able to nearly double the percentage of correct values, also in a challenging low quality dataset.

It is important to notice that `vat number` and `total` are the elements with the greatest correctness increment in both datasets. The `vat number` structure follows certain constrains, and the great correctness improvement (44.2% in average) confirms the usefulness of semantic checks; similarly the correctness improvement of the `total` element (40.7% in average) confirms the usefulness of joint semantic checks.

## CONCLUSIONS

In this work we consider the problem of detecting and correcting OCR errors in an information extraction system. We propose an approach based on domain-knowledge about the possible misrecognition of characters to suggest corrections. We also propose a set of semantic and syntactic checks to validate the suggested corrections.

The system here proposed is able to nearly double the percentage of correctly identified values. These results have been obtained with a real-world dataset including a substantial amount of noise, as typically occurs when digitalizing printed documents previously handled by a corporate office. Despite the challenging dataset used in our experimental evaluation, the obtained results are highly encouraging.

Future work will be devoted to extending further the generation of the corrections, possibly for allowing an automatic definition of the substitution rules.

## ACKNOWLEDGMENT

We are very grateful to Matteo Gazzin for his hard work during the initial stage of this research.

## REFERENCES

[1] A. Amano, N. Asada, M. Mukunoki, and M. Aoyama. Table form document analysis based on the document structure grammar. *International Journal on Document Analysis and Recognition*, 8(2):201–213, June 2006.

[2] S. M. Beitzel, E. C. Jensen, and D. A. Grossman. Retrieving ocr text: A survey of current approaches. In *Symposium on Document Image Understanding Technologies (SDUIT*, 2003.

[3] Y. Belaid and A. Belaid. Morphological tagging approach in document analysis of invoices. In *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 1 - Volume 01*, pages 469–472. IEEE Computer Society, 2004.

[4] F. Cesarini, E. Francesconi, M. Gori, and G. Soda. Analysis and understanding of multi-class invoices. *International Journal on Document Analysis and Recognition*, 6(2):102–114, Oct. 2003.

[5] A. Fujiyoshi, M. Suzuki, and S. Uchida. Syntactic detection and correction of misrecognitions in mathematical ocr. In *Proceedings of the 2009 10th International Conference on Document Analysis and Recognition*, ICDAR '09, pages 1360–1364, Washington, DC, USA, 2009. IEEE Computer Society.

[6] J. J. Hull and S. N. Srihari. Experiments in text recognition with binary n-gram and viterbi algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 4:520–530, May 1982.

[7] K. Kukich. Techniques for automatically correcting words in text. *ACM Comput. Surv.*, 24:377–439, December 1992.

[8] E. Medvet, A. Bartoli, and G. Davanzo. A probabilistic approach to printed document understanding. *International Journal on Document Analysis and Recognition IJDAR*, 2010.

[9] D. Miller, S. Boisen, R. Schwartz, R. Stone, and R. Weischedel. Named entity extraction from noisy input: Speech and ocr. In *In Proceedings of the 6th Applied Natural Language Processing Conference*, pages 316–324, 2000.

[10] Y. Navon, E. Barkan, and B. Ophir. A generic form processing approach for large variant templates. In *Proceedings of the 2009 10th International Conference on Document Analysis and Recognition*, ICDAR '09, pages 311–315, Washington, DC, USA, 2009. IEEE Computer Society.

[11] K. Taghva, R. Beckley, and J. Coombs. The effects of ocr error on the extraction of private information. In H. Bunke and A. Spitz, editors, *Document Analysis Systems VII*, volume 3872 of *Lecture Notes in Computer Science*, pages 348–357. Springer Berlin / Heidelberg, 2006. 10.1007/11669487_31.

[12] M. L. Wick, M. G. Ross, and E. G. Learned-Miller. Context-sensitive error correction: Using topic models to improve ocr. In *ICDAR'07*, pages 1168–1172, 2007.

[13] Wikipedia. Codice fiscale — Wikipedia, the free encyclopedia, 2011. [Online; accessed 22-december-2011].