IJDAR (2011) 14:335–347
DOI 10.1007/s10032-010-0137-1

ORIGINAL PAPER

A probabilistic approach to printed document understanding

Eric Medvet · Alberto Bartoli · Giorgio Davanzo

Received: 8 January 2010 / Revised: 14 July 2010 / Accepted: 19 October 2010 / Published online: 5 November 2010

Abstract We propose an approach for information extraction for multi-page printed document understanding. The approach is designed for scenarios in which the set of possible document classes, i.e., documents sharing similar content and layout, is large and may evolve over time. Describing a new class is a very simple task: the operator merely provides a few samples and then, by means of a GUI, clicks on the OCR-generated blocks of a document containing the information to be extracted. Our approach is based on probability: we derived a general form for the probability that a sequence of blocks contains the searched information. We estimate the parameters for a new class by applying the maximum likelihood method to the samples of the class. All these parameters depend only on block properties that can be extracted automatically from the operator actions on the GUI. Processing a document of a given class consists in finding the sequence of blocks, which maximizes the corresponding probability for that class. We evaluated experimentally our proposal using 807 multi-page printed documents of different domains (invoices, patents, data-sheets), obtaining very good results—e.g., a success rate often greater than 90% even for classes with just two samples.

Keywords Document understanding · Automatic model upgrading · Invoice analysis · Maximum likelihood

E. Medvet (✉) · A. Bartoli · G. Davanzo
DEEI, University of Trieste, Via A. Valerio 10,
34127 Trieste, Italy
e-mail: emedvet@units.it

A. Bartoli
e-mail: bartoli.alberto@units.it

G. Davanzo
e-mail: giorgio.davanzo@gmail.com

1 Introduction

Information extraction from printed documents plays a key role in many areas: office automation, knowledge management, intelligence and so on. Manual processing is often not feasible or fully satisfactory, typically because of the high volume of documents to be processed and the high cost per-unit introduced by human operators. Augmenting the automation degree of information extraction from printed documents may thus have a substantial impact on many settings of highly practical interest.

Achieving full automation is still a challenge, due to the huge variability of content types and layouts. The involvement of human operators is hence required, in particular for providing a description of logical structures of documents, hereinafter *models*, which systems use for extracting information from such documents. Models may be described either explicitly or by means of a training set of suitably annotated sample documents. A model effectiveness is usually bounded to documents of the same *class*, i.e., documents that share layout and content type such as invoices from a given firm or data-sheets for a given product category. A human operator is required whenever a new class is to be handled, in order to define the corresponding model. Clearly, the amount of work and skill required for defining a new model should be kept as small as possible, especially when the occurrence of new classes may be a frequent event or the number of documents of a given class may be small.

In this paper, we propose an approach for information extraction from multi-page printed documents that has very good performance and is designed to make the definition of a new model a very simple and lightweight operation. When a new model has to be defined, a training set composed of one or more documents of the new class is required. For each document in the training set, the system automatically

transforms the document into a set of OCR-generated blocks, and all the operator has to do is selecting the block sequence containing the piece of information to be extracted (clearly, the system is capable of extracting multiple pieces of information from a given document, but this point is irrelevant to the discussion). The operator thus merely chooses from a set of blocks drawn over a picture of the document, which may be done without any specialized skill and with a few mouse clicks using a very simple GUI.

We represent models in terms of values for attributes of OCR-generated blocks that pertain to both layout and content, e.g., horizontal/vertical position, page number, text length and alike. During model construction, we use the attribute values extracted from the blocks selected by the operator to fit several univariate probability distributions—one for each attribute—according to the maximum likelihood method: for each distribution, we choose the values of its parameters in order to maximize the probability of the observed attribute value on the documents in the training set. The resulting parameters constitute the internal representation of a model. Given a new document of a class whose model is known (we remark that the matching of documents with classes is beyond the scope of this work—see also Sect. 2), we identify the block sequence containing the information to be extracted by simply selecting the sequence that maximizes the corresponding probability—i.e., the one that is most probable given the model. The practical application of each of these steps requires few trivial computations.

In general, the probability that a given block sequence contains the required information follows a multivariate distribution of the attributes of the corresponding blocks. As will be clearer from the paper, the resulting expression is very complex and not practically useful. The fact that some attributes are textual increases complexity further. An important component of our contribution consists in the derivation of a more tractable expression, based on univariate distributions and amenable to fitting by means of the maximum likelihood method.

We evaluated the performance of our proposal on a dataset composed by 807 multi-page printed documents belonging to three radically different domains: invoices, patents and electronic components data-sheets. The experimental results show that our proposal is very effective, even when the training set is composed of very few samples. The ability to handle effectively a very small training set allows keeping human involvement to a minimum and is a necessity whenever the arrival rate of documents belonging to the new model is very low. We also propose and evaluate a method for automatically upgrading a model definition, as more and more documents of the model are processed.

The remaining part of this paper is organized as follows. Section 2 presents an overview about current state of the art in the field. In Sect. 3, we present the scenario and our approach from a high level point of view: in particular, we introduce the model, which describes how the information should be extracted from a document of a given class. In Sect. 4, we show how we derive a general form for the probability distribution, the main component of the model, explaining the base assumptions. In Sect. 5, the details about how we build a model from a training set, along with a synthetic example of the procedure, are provided. Section 6 presents our experimental evaluation, and Sect. 7 summarizes this work conclusions.

2 Related work

Document understanding is a process that consists in analyzing documents in order to extract human understandable information and codify it into machine-readable form. The complexity of document understanding and its system engineering challenges, as well as insights into important works in this field, is discussed in [10].

Information extraction systems can be classified depending on whether the class of the document to be processed is known or unknown. When the system does not need to know the document class, a fair amount of a priori knowledge about the specific application domain—i.e., invoices, medical receipts and so on—is usually necessary and embedded within the system. For example, a table of "main primary tags" is used in [5] to identify labels of interesting information. This table utility is bounded to the domain of invoices and, perhaps more importantly, is language-dependent. A similar solution is followed by [11] where certain words are marked as keywords. Our approach does not depend on such a priori knowledge, which allows us to address different application domains and languages equally well and, most importantly, without any domain- or language-specific precompilation.

Systems where the document class is known—like ours—are generally more effective at identifying and extracting the desired information, but they have to face the problems of (i) associating each document with the corresponding class and (ii) defining a document model for each class. Both issues require a human operator, with varying degrees of involvement and required skills depending on the specific system. The matching of documents with classes is beyond the scope of this work: in our system, we use an automated open world classifier based on the spatial density of black pixels and of image edges, full details can be found in [4] and [20]. In general, there is a fair amount of research about the problem of matching documents against a predefined set of stored templates. An interesting approach involves the notion of component block representation, similar to our block level description of a document, and is presented in [17]. Insights into document classification in this context can be found in [6,9,12].

Definition of document models usually requires an operator to identify labels and locations for all metadata of the corresponding class [15]. This operation is performed through a specialized GUI, and some amount of specific skill is required. This approach is followed, essentially, by most of the commercial solutions currently available, which require an operator to "draw" a model for the given class. In our work, we attempt to keep the model definition phase as simple and lightweight as possible, in particular by not requiring any specific IT-related skill—it seems fair to claim that any administrative operator will have the ability to identify the OCR-generated blocks containing the required information and click on them.

The approach presented in [7] is based on a learning procedure similar to the systems mentioned above, as it requires a table (file) where logical objects of interest and related tags have been manually located on a number of sample invoices of the given class. The impact of the training set size on the approach effectiveness is not investigated. In our work, we assessed the performance of our approach with varying size for the training set and found very good performance even with very small training sets—in the order of 2–3 documents. We also proposed and evaluated different policies for updating the system knowledge during its usage: we show experimentally that performance may improve by letting the system learn from its processing results, even without requiring feedback from operators. Our dataset is a superset of the publicly available dataset used by [7]. We found that our approach tends to perform better, although we could not perform a detailed quantitative comparison (see Sect. 6 for details). We attribute this tendency to the fact that our approach may successfully identify blocks whose textual content has been garbled by the OCR system, whereas OCR errors simply prevented the system in [7] from identifying the searched tags or labels.

The fact that our approach does not depend on the ability to identify any specific label or keyword indeed implies that OCR performance is not critical for the quality of the results. As will be clearer from the following sections, we identify relevant blocks basing on their probability of being consistent with some geometric properties (size, position, page) and textual information (text length, text alignment, content) as specified by the model. It follows that even when the OCR fails to detect correctly such pieces of text as `"Total"` or `"Price"`, our system generally is still able to identify the relevant information. Of course, OCR performance remains crucial for correctness of the information actually extracted—several workarounds may be used in practice to mitigate this issue, though (e.g., type checks, range checks and alike).

We have not performed a quantitative analysis of the impact of OCR performance on the quality of information location and extraction. Indeed, this is a topic where further research is needed even from a methodological point of view [13,16]. On a qualitative level, however, many of the documents in our dataset were of bad quality and resulted in bad OCR results.

An approach that describes documents in terms of attributed relational graphs is presented in [8]: an excellent effectiveness in detecting the searched information is reported, but the experimental setting is limited to two classes. A similar method is proposed later in [2]: in order to expand the coverage of the graphs-based approach, the authors rely on statistical decision trees and bi- and tri-grams applied to the block textual contents. The study aims at recognizing the document structure, i.e., at identifying general information (e.g., title, body, captions, …) of textual documents. The model consists of a set of geometric and logical structures and is based on common-sense reasoning and statistical methods. The proposed method effectiveness is assessed on a dataset of about 800 single-page documents, whose complexity is set on three levels basing on the number of objects—which roughly corresponds to text areas—in the document. Another method for document structure recognition is proposed in [14]: the authors build fuzzy logical rules for block classification that involve considering both layout and textual features. Similar to our work, an operator can define a rule by providing a positive example by means of a GUI and some mouse clicks; yet, the author do not investigate on their system ability to learn from a training set composed of multiple documents.

A problem that is not closely related to information extraction from printed documents, but which also relies on document analysis, is the identification of reading order. The system proposed in [1,21] extracts several features from color scanned documents written in English (e.g., coordinates, foreground and background color, font size and type,…) and then uses spatial inference and natural language processing in order to capture the relationship among blocks of printed text.

Many of the studies focusing on printed document understanding deal with invoices [5,7,8,11] and forms [3,18], due to the economic relevance of these kinds of documents in terms of costs and volumes. In this respect, the processing cost of a printed invoice has been estimated in about $ 13 per document, and [18] reports that the number of printed paper forms that are handled yearly by Japanese public administration is greater than 2 billions. In our work, we propose an approach that is not tailored to a specific type of document, and we evaluated it on a dataset composed of documents from radically different application domains: not only invoices, but also patents and data-sheets.

Concerning printed forms, their automatic processing is the focus of [3]. The authors propose a document structure grammar that can be represented in a Table Form Markup Language (TFML): a semi-automatic procedure is described to analyze an empty form layout and describes its

structure in terms of TFML. The procedure heavily relies on printed rules, which are not always present in other kinds of documents—e.g., those in our dataset: patents, data-sheets, invoices.

Although the problems involved in automated processing of invoices and forms are largely similar, the invoice analysis problem is perhaps more challenging, because invoice layouts exhibit a large diversity, due to the huge number of firms that emit invoices. Invoice analysis systems need to address the model definition phase, which is especially critical when the occurrence of new models is a frequent event. The authors of [19] quantify the economic impact of accountant personnel workload involved in building and maintaining invoice models for a widely adopted invoice recognition software. The model definition phase is central in our work, and we also propose a method for automatically upgrading a model and thus mitigating the model maintenance cost.

3 Our approach overview

3.1 Scenario

A *document* D is represented as a set of block, denoted by $\{b_1, b_2, \dots\}$. A *block* consists of a position, a size and a content. The position is specified by the page number p and the coordinates x and y from the page origin (upper left corner). The size is specified by width w and height h. The content is specified by the single-line textual content l, expressed as a character sequence. A block b is hence defined as a tuple of *block attributes* $b = \langle p, x, y, w, h, l \rangle$. See Fig. 1 for an example of blocks with depicted attributes.

An OCR system may be used to obtain a representation of this kind from a physical document. Different OCR systems may lead to different representations for the same physical document. The details of this procedure are irrelevant to this discussion.

A document is associated with a *schema* and a document model. Intuitively, the former describes the information to be extracted from the document, whereas the latter describes how to identify that information. Documents of the same class, e.g., invoices produced by the same firm, share the same model.

More in detail, the schema consists of a set of typed *elements*. The understanding of a document corresponds to finding a *value* for these elements, based on the content of the document, i.e., of its blocks. From a different point of view, a document corresponds to a row of a table whose columns are specified by the corresponding schema. For example, a schema could be given by `Date`, `Total Amount`, `Document Number` and so on (we omit the corresponding types for brevity).

The document model, *model* for short, consists of a set of *rules*, exactly one rule for each element of the schema. A rule describes how to extract the corresponding value from the document blocks. More details about the rules and specific examples will be given below.

We remark that the correspondence between values and blocks is not one-to-one. More in details, the following conditions may or may not occur independently: (a) the value is contained in a single block; (b) no other information is contained in the block/blocks other than the value—or the portion of it. For an example, see Sect. 3.2.

By understanding of a document, we mean the process of extracting from the document values for all items of the

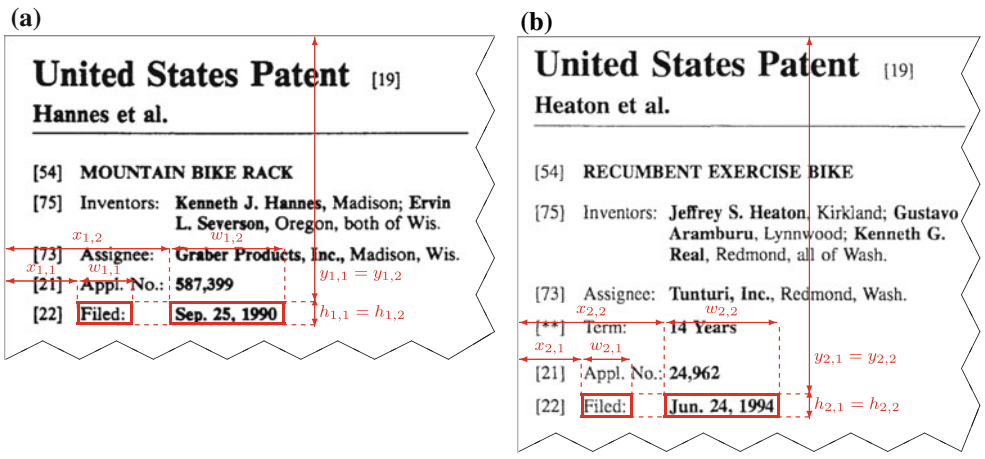

Fig. 1 The portions of two documents of our dataset, belonging to the same class in the `Patents` group. The corresponding true sequences for the `Filing Date` element are highlighted, and the attributes for the corresponding blocks are depicted (see Sect. 5). Note that a different number of text rows lays above the considered element in the two documents

schema associated with the document. Our contribution consists of:

1. a procedure for understanding a document automatically, given a model;
2. a procedure for constructing a model automatically, given a set of documents.

3.2 Rule

Each rule of a document model is associated with one and only one schema element e. A rule is composed of a *cardinality*, a *matching probability* and an *extraction function*. The cardinality n_e defines the length of the sequence of blocks, say $B = \langle b_1, \ldots, b_{n_e} \rangle$, to which the rule may be applied. The matching probability $\hat{P}_e(B)$ is the probability that B contains a value for element e. The extraction function $f_e(B)$ is a function for computing that value as $v = f_e(B)$.

For example, the rule for the `Date` element could be composed of a matching probability which, informally, encodes this description:

> the `Date` is contained in a pair of blocks: the first stays on the first page, at about 2.2 cm from left page margin and 4 cm from top page margin, with a content similar to `"Day: 02/03"`; the second stays on the same page, at about 2 cm left of the first one and 0.5 cm lower, with a content in the form `"Year: 2009"`

The extraction function for the same rule could, informally, say:

> extract and concatenate the block contents which follow the colon.

The cardinality for this example rule is $n_e = 2$.

The understanding of a document consists of the two following steps, for each rule $\langle n_e, \hat{P}_e, f_e \rangle$ in the model:

1. find the *matching sequence* B^*, i.e., the n_e-length sequence of blocks, which maximizes $\hat{P}_e$;
2. extract the value v from the matching sequence as $v = f_e(B^*)$.

The building of a model consists of generating a set of rules given a set of documents: for each element, a rule is generated starting from the corresponding values. Such procedure is described in Sect. 5.

A crucial component of our contribution consists in expressing the matching probability $\hat{P}_e$ in a tractable form and in finding a simple method for estimating its numerous parameters. More in detail, we want to express the matching probability, which concerns rather complex events (i.e., the event that a sequence of blocks contains a given value), as a function of simple univariate probability distributions of independent random variables obtained from the block attributes (i.e., p, x, y, w, h, l). The details about the way in which we derived the simple form for $\hat{P}_e$ are given in Sect. 4.

The extraction function operates on the concatenation of the textual contents of the blocks of the matching sequence. In general, the value can be extracted using a regular expression depending largely only on the given element. Yet, while the matching sequence could be found even in the presence of OCR errors, because of such errors, some refinement could be necessary in order to extract the correct value itself.

4 Matching probability details

In this section, we describe how we derived a simpler form for the matching probability $\hat{P}_e$ (we will omit the e subscript for brevity). All the following reasoning applies to the document understanding task itself within the scenario described above and is not bounded to a specific category of documents.

We estimate the probability $\hat{P}(B)$ for a sequence $B = \langle b_1, \ldots, b_n \rangle$ of n blocks as follows:

$$\hat{P}(B) = \prod_{i=1}^{n} P_i(b'_i) = \prod_{i=1}^{n} P_i(\langle p'_i, x'_i, y'_i, w_i, h_i, l_i \rangle) \qquad (1)$$

where $p'_i = p_i$ when $i = 1$ and $p'_i = p_i - p_{i-1}$ otherwise, $x'_i = x_i$ when $i = 1$ or $p_i \neq p_{i-1}$ and $x'_i = x_i - x_{i-1}$ otherwise, and the same for y'_i. In other words, we express the position attributes of b_i relatively to the position attributes of b_{i-1} : p'_i, x'_i, y'_i is hence the relative position of the i-th block.

The rationale for this approximation is as follows. First, we assume that the size and content attributes of a block do not depend on any of the other blocks attributes. Second, we assume that the *relative* position of b_i does not depend on the position—neither absolute, nor relative—of any other block, although the *absolute* position of b_i is in general dependent on the absolute position of b_{i-1}. It follows that the probability that both *dependent* events "b_{i-1} is the $i-1$-th block of the matching sequence" and "b_i is the i-th block of the matching sequence" occur is equal to the probability that both the *independent* events "b'_{i-1} is the $i-1$-th block of the matching sequence" and "b'_i is the i-th block of the matching sequence" occur. For example, consider a rule for sequences of 2 blocks for the `Total` element of invoices, the first being the label and the second the actual value. If on a given document, the first block is lower on the page than the usual, possibly for the presence of many item lines in the invoice, the second block will also be lower, but, in general, at the same distance from the first block: that is, y_2 depends on y_1, whereas $y'_2 = y_2 - y_1$ does not.

By the aforementioned assumptions, $\hat{P}$ can simply be expressed as a product of probabilities P_i related to single

blocks. Each P_i, which we call the *block probability*, concerns a single block that is a tuple composed of 6 attributes whose corresponding 6 random variables are in general dependent on each other. We identified a domain knowledge-based set of dependencies, which allowed us to elaborate and simplify the form of P_i, as follows.

First, we can use marginalization in order to write P_i basing on the possible values for the page p'_i:

$$P_i(b'_i) = \sum_k P(b'_i \cap p'_i = k)$$

where $P(b_1 \cap p'_i = k)$ is the joint probability of the following two events: b'_i is the i-th block of the matching sequence and $p'_i = k$. These two events are in general dependent. For example, consider an invoice class, where the total amount value may be located at the bottom of the first page or at the top of the second page: it follows that low values for y are more probable, if the page attribute is equal to 2, and great values for y are more probable, if the page attribute is equal to 1—in other words, the y attribute of the block is dependent on the p attribute. We can rewrite the joint probability in terms of conditional probability on the page p'_i:

$$\begin{aligned} P(b'_i \cap p'_i = k) &= P(b'_i | p'_i = k) P(p'_i = k) \\ &= P_{i,k}(\langle x'_i, y'_i, w_i, h_i, l_i \rangle) P(p'_i = k) \end{aligned}$$

where $P_{i,k}(\langle x'_i, y'_i, w_i, h_i, l_i \rangle)$ is the probability that a block identified by the tuple $\langle x'_i, y'_i, w_i, h_i, l_i \rangle$ is the i-th block of the matching sequence, given that its page p'_i is equal to k. Concerning $P(p'_i = k)$, we assume that there is a finite set $K_i = \{k_1, k_2, \dots\}$ of possible values for p'_i, whose corresponding probabilities are $s_{i,k_1}, s_{i,k_2}, \dots$. In other words, $P(p'_i = k) = s_{i,k} I(p'_i; k)$, where $I(p'_i; k) = 1$ for $p'_i = k$ and 0 otherwise and $s_{i,k} = 0$ if $k \notin K_i$. Thereby, we can write P_i as follows:

$$P_i(b_i) = \sum_k s_{i,k} I(p'_i; k) P_{i,k}(\langle x'_i, y'_i, w_i, h_i, l_i \rangle)$$

Concerning $P_{i,k}(\langle x'_i, y'_i, w_i, h_i, l_i \rangle)$, we assume that y'_i and h_i are independent from the other three variables. In particular, note that, since blocks contain exactly one line of text, the height attribute h_i is largely independent from its text content l_i. Hence, we can write:

$$P_{i,k}(\langle x'_i, y'_i, w_i, h_i, l_i \rangle) = P^{y'}_{i,k}(y'_i) P^{h}_{i,k}(h_i) P^{x',w,l}_{i,k}(\langle x', w, l \rangle) \quad (2)$$

Then, we split the x'_i, w_i, l_i dependency in one between x'_i and w_i and another between w_i and the text content l_i. The dependency between x'_i and w_i represents the fact that a given text could be aligned in three different ways: left, center or right (justified text may be handled in any of these three cases, for the purpose of this analysis). It follows that:

- in case of left-alignment, x'_i and w_i are independent;
- in case of center-alignment, $x^c_i = x' + \frac{w_i}{2}$ and w_i are independent;
- in case of right-alignment, $x^r_i = x'_i + w_i$ and w_i are independent.

The dependency between w_i and l_i represents the fact that, in general, the longer the text content, the larger the block width. We define $w'_i = \frac{w_i}{\mathcal{L}(l_i)}$ as the average width of the characters composing the block text content, being $\mathcal{L}(l_i)$ the number of characters in l_i: we assume that w'_i and l_i are largely independent, since w'_i depends on the font size and type, rather than on the text content.

We can hence write $P^{x',w,l}_{i,k}(\langle x'_i, w_i, l_i \rangle)$ in three possible forms depending on text alignment:

$$P^{x',w,l}_{i,k}(\langle x', w, l \rangle) = \begin{cases} P^{x'}_{i,k}(x'_i) P^{w'}_{i,k}(w'_i) P^{l}_{i,k}(l_i), & \text{left} \\ P^{x^c}_{i,k}(x^c_i) P^{w'}_{i,k}(w'_i) P^{l}_{i,k}(l_i), & \text{center} \\ P^{x^r}_{i,k}(x^r_i) P^{w'}_{i,k}(w'_i) P^{l}_{i,k}(l_i), & \text{right} \end{cases}$$

which can be summarized as:

$$P^{x',w,l}_{i,k}(\langle x', w, l \rangle) = P^{x''}_{i,k}(x''_i) P^{w'}_{i,k}(w'_i) P^{l}_{i,k}(l_i)$$

where x''_i is a shortcut symbol that represents one among x'_i, x^c_i and x^r_i.

Finally, we obtain the following general form for the matching probability:

$$\hat{P}(B) = \prod_{i=1}^{n} \left(\sum_k s_{i,k} I(p'_i; k) P^{y'}_{i,k}(y'_i) P^{h}_{i,k}(h_i) \times P^{x''}_{i,k}(x''_i) P^{w'}_{i,k}(w'_i) P^{l}_{i,k}(l_i) \right) \quad (3)$$

Note that each P is a univariate distribution.

Next, we assume that the size attributes (w' and h) and the position attributes (x'' and y') can be described as random variables with normal distribution (denoted by $N(\mu, \sigma)$). In a preliminary phase of our study, we considered using other distributions for the aforementioned random variables—in particular the Uniform Distribution—but we found the normal distribution models them better.

Concerning the textual content, $P^{l}_{i,k}(l_i)$ is the *text probability* that the text l_i is the text of the i-th block of the matching sequence; $P^{l}_{i,k}$ hence operates on text, differently from all other probabilities that operate on numbers. We assume that $P^{l}_{i,k}(l_i)$ can be expressed as a Markov chain of order 2. Its state space corresponds to the set of possible characters and 2 pseudo-characters representing the begin and end of the text. The probability M of the text l is defined as the probability of the state sequence corresponding to l. For example, the probability of the word `"two"` is given by:

Table 1 Parameters of the matching probability of a rule

Parameters	Meaning
$s_{i,k}$	Probability of $p'_i = k$
$\mu^{x''}_{i,k}$	Mean for the adjusted relative x position (x'')
$\sigma^{x''}_{i,k}$	Scale for the adjusted relative x position (x'')
$\mu^{y'}_{i,k}$	Mean for the relative y position (y')
$\sigma^{y'}_{i,k}$	Scale for the relative y position (y')
$\mu^{w'}_{i,k}$	Mean for the character width w'
$\sigma^{w'}_{i,k}$	Scale for the character width w'
$\mu^{h}_{i,k}$	Mean for the height h
$\sigma^{h}_{i,k}$	Scale for the height h
$T_{i,k}$	Transition matrix for text probability

Each parameter concerns the i-th block. All parameters are numerical except for the last one, which is a matrix

$$M(\texttt{"two"}) = P(\texttt{"}\triangleright\texttt{t"}|\texttt{"}\triangleright\texttt{"})P(\texttt{"tw"}|\texttt{"}\triangleright\texttt{t"})$$
$$P(\texttt{"wo"}|\texttt{"tw"})P(\texttt{"o}\triangleleft\texttt{"}|\texttt{"wo"})$$

where $\triangleright$ and $\triangleleft$ represent the begin and the end of the text, respectively, and each transition probability corresponds to an element of the transition matrix $T_{i,k}$. The details about how we set $T_{i,k}$ are given in Sect. 5.

At the end, the final form for the matching probability is the following:

$$\hat{P} \equiv \prod_{i=1}^{n}\left(\sum_k s_{i,k} I(k) N(\mu^{y'}_{i,k}, \sigma^{y'}_{i,k}) N(\mu^{h}_{i,k}, \sigma^{h}_{i,k}) \times N(\mu^{x''}_{i,k}, \sigma^{x''}_{i,k}) N(\mu^{w'}_{i,k}, \sigma^{w'}_{i,k}) M_{i,k}\right) \quad (4)$$

where we omitted the function arguments for readability.

In summary, defining a new model merely translates in choosing a suitable value for the parameters of the above formula (and for the cardinality n, i.e., the number of blocks) that we summarize in Table 1. In the next section, we describe how we choose these values.

5 Model building

We describe here the procedure for defining a model. The operator provides a set of documents and defines the schema of the new model being defined. For each document d, the system processes d by means of an OCR and presents to the operator a graphical representation of d in terms of its composing blocks. The operator then selects on this representation, for each element of the schema, the sequence of blocks containing the value of the element. An example of the result of this action for a document, for a single element, is shown in Fig. 1. We call this sequence the *true sequence*. The operator might also specify that d does not contain a value for a given element. We require that the true sequences corresponding to the same element of the schema have the same length for each d: if this requirement is not met, a work-around for this limitation exists and consists in partitioning the training set in subsets in which each sequence of blocks has the same size. Once all documents in the set have been analyzed, the system processes the true sequences provided by the operator as described below. We will describe the procedure for one single element, since the rule for an element can be generated independently from rules for the other elements.

Concerning the rule cardinality n, we set it to the length of the true sequences—this value is either 1 or 2 in our experiments. Concerning the extraction function f, its building consists of choosing the most suitable candidate among a predefined set of extraction functions suitable for the given element. For example, consider the `Date` element, the set could be composed of few different regular expression (see Sect. 3.2) corresponding to different date formats. We will not further elaborate on this issue.

The interesting part concerns the matching probability $\hat{P}$. The goal of the procedure consists in choosing a value for all parameters in Table 1. The choice is made in order to fit the distributions in Eqn. 4 according to the maximum likelihood method. As will be clear, the actual processing is quite simple and may be embedded in a model-independent GUI easily.

Let D be the training set composed of the m true sequences that the operator provides as described before and let $B^*_i = \langle b^*_{i,1}, \ldots, b^*_{i,n}\rangle$ be the i-th true sequence. We use the maximum likelihood method, as follows.

- We set $s_{i,k}$ to the frequency of i-th blocks in D whose relative page p'^*_i is k.
- For each i, k-th Normal Distribution of Eq. 4, we estimate its $\mu_{i,k}$ and $\sigma_{i,k}$ parameters, using the corresponding attributes of i-th blocks in D whose relative page p'^*_i is k. In order to avoid building a too specific model for the considered attributes, we impose a lower bound for the σ parameter, which we denote with σ_{xy} for the position attributes and σ_{wh} for the size attributes.
- For each i and k, we choose the x'_i, w_i, l_i dependency that maximizes the probability of i-th blocks in D whose relative page p'^*_i is k.

Concerning the text probability $M_{i,k}$, we proceed as follows. Before actually processing a text l, we perform some simple elaboration, which consists of: (i) transform to lowercase, (ii) replace all digit characters with `"#"`, (iii) replace all space characters with standard space character, (iv) replace all punctuation characters with `"."` and finally (v) replace any other character with `"*"`. We denote the result with l'. Recall that we can describe a chain of order two using a transition matrix T of size $a \times a^2$, being a the number of states. In our case, given the aforementioned text elaborations, $a = 32$ and T indexes represent, respectively,

a single character and a sequence of two characters: e.g., $t_{3,34} = t_{\texttt{"c"},\texttt{"ab"}} = P(\texttt{"bc"}|\texttt{"ab"})$. In order to set the $T_{i,k}$ matrix for $M_{i,k}$, we start with a frequency matrix F with the same size of $T_{i,k}$ and each element set to 0. For example, after processing the sequence `"banana"`, we will have $f_{\texttt{"a"},\texttt{"▷b"}} = f_{\texttt{"n"},\texttt{"ba"}} = f_{\texttt{"◁"},\texttt{"na"}} = 1$ and $f_{\texttt{"a"},\texttt{"an"}} = 2$. Then, we process each true sequence textual content $l_i'^*$ and increment the corresponding F elements. At the end, we set for each pair of indexes u, v:

$$t_{u,v} = \begin{cases} (1-\epsilon)\frac{f_{u,v}}{\sum_{z=1}^{a^2} f_{u,z}}, & \text{if } f_{u,v} > 0 \\ \frac{\epsilon}{a^2 - N_u}, & \text{otherwise} \end{cases} \quad (5)$$

where N_u is the number of $f_{u,v}$ that is greater than 0. We use the term ϵ to make the text probability smoother, i.e., such that it assigns non-zero (yet low) probabilities also to textual contents that are not present in the training set.

We set our parameters as follows: $\sigma_{xy} = \sigma_{wh} = 0.05$ inches $= 1.27$mm, $\epsilon = \frac{1}{3}$.

5.1 Model building example

We provide here an example of building a model for one element. Figure 1 shows the top-right portions of two documents of our dataset. The documents belong to the same class and represent patents (see next section for a description of our dataset). The true sequences for the `Filing Date` element, as selected by the operator, are highlighted. For each document, the operator simply selected, by means of a GUI, two blocks that were already detected by the OCR system.

In this example, the training set D is composed of $m = 2$ documents; thus, there are two true sequences composed of two blocks each: $B_1^* = \langle b_{1,1}^*, b_{1,2}^* \rangle$ and $B_2^* = \langle b_{2,1}^*, b_{2,2}^* \rangle$. Values for blocks attributes follow (where applicable, values are expressed in mm):

	p	x	y	w	h	l
$b_{1,1}^*$ = ⟨	1,	12.5,	48.7,	8.6,	3.1,	`"Filed:"` ⟩
$b_{1,2}^*$ = ⟨	1,	28.2,	48.7,	18.8,	3.1,	`"Sep. 25, 1990"` ⟩
$b_{2,1}^*$ = ⟨	1,	11.0,	64.4,	7.8,	3.1,	`"Filed:"` ⟩
$b_{2,2}^*$ = ⟨	1,	25.1,	64.4,	18.8,	3.1,	`"Jun. 24, 1994"` ⟩

Rule cardinality is set to $n = 2$. The values for the variables derived as explained in Sect. 4 are hence computed by the system as:

	p'	x''	y'	w'	h	l'
$b_{1,1}'^*$ = ⟨	1,	12.5,	48.7,	1.43,	3.1,	`"Filed."` ⟩
$b_{1,2}'^*$ = ⟨	0,	15.7,	0.0,	1.45,	3.1,	`"Sep. ##. ####"` ⟩
$b_{2,1}'^*$ = ⟨	1,	11.0,	64.4,	1.30,	3.1,	`"Filed."` ⟩
$b_{2,2}'^*$ = ⟨	0,	14.1,	0.0,	1.45,	3.1,	`"Jun. ##. ####"` ⟩

At this point, all parameters for the matching probability—see Table 1—can be computed. Concerning the page attribute, we have:

$$s_{1,k} = \begin{cases} 1, & \text{if } k = 1 \\ 0, & \text{otherwise} \end{cases}$$

$$s_{2,k} = \begin{cases} 1, & \text{if } k = 0 \\ 0, & \text{otherwise} \end{cases}$$

For the other parameters, only the values for $(i = 1 \wedge k = 1)$ or $(i = 2 \wedge k = 0)$ are needed, since all the blocks of the true sequences lay on the first page. These values are as follow (we omit the transition matrix for the sake of brevity):

$\mu_{1,1}^{x''} = 11.75$	$\mu_{2,0}^{x''} = 14.9$
$\sigma_{1,1}^{x''} = 1.27^{\dagger}$	$\sigma_{1,1}^{x''} = 1.27^{\dagger}$
$\mu_{1,1}^{y'} = 56.55$	$\mu_{2,0}^{y'} = 0$
$\sigma_{1,1}^{y'} = 7.85$	$\sigma_{2,0}^{y'} = 1.27^{\dagger}$
$\mu_{1,1}^{w'} = 1.36$	$\mu_{2,0}^{w'} = 1.45$
$\sigma_{1,1}^{w'} = 1.27^{\dagger}$	$\sigma_{2,0}^{w'} = 1.27^{\dagger}$
$\mu_{1,1}^{h} = 3.1$	$\mu_{2,0}^{h} = 3.1$
$\sigma_{1,1}^{h} = 1.27^{\dagger}$	$\sigma_{2,0}^{h} = 1.27^{\dagger}$

The values denoted by the symbol † are obtained by lower-bounding the corresponding computed σ estimates with the proper lower bound (i.e., σ_{xy} or σ_{wh}).

6 Experiments

6.1 Dataset

In order to assess our approach effectiveness, we collected a real-world dataset composed of 4 *groups* of documents, totaling 807 multi-page documents divided in 85 classes. We partitioned the dataset basing on different document domains (invoices, patents or data-sheets) or, in case of `Invoice-1` and `Invoice-2`, basing on the dataset source.

The first group—which we call `Invoices-1`—is composed of 406 multi-page invoices, issued by 32 different businesses, each issuer corresponding to a different class. The largest class contains 79 invoices; the first 15 largest classes account for about 80% of this group documents.

The second group—`Invoices-2`—is a subset of the dataset used by [7] and is composed of 156 single-page invoices, belonging to 33 classes. With respect to the original dataset used by the authors of the cited paper, we discarded the classes containing only one document. The largest class contains 25 documents, and the first 10 largest classes account for about 58% of this group documents.

The third group—`Patents`—is composed of 135 patents obtained from 10 different patents sources, each patent source corresponding to a different class. The largest class contains 22 patents, and 7 classes contain 10 or more patents.

The last group—`Data-sheets`—is composed of 110 data-sheets of electronic components (e.g., Zener diodes) divided in 10 classes. Data-sheets of the same class share

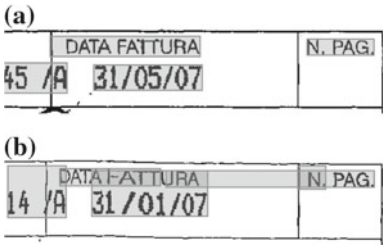

Fig. 2 Portions of two different documents of a class of `Invoices-1`: the *shaded rectangles* represent the blocks. The document on the right shows a segmentation error, which also caused the textual content to wrongly be recognized as `"31/OI/0?"`

the producer and the component type. The 5 largest classes account for about 85% of this group documents.

Documents of the first two groups have been obtained by scanning the corresponding paper documents as black-and-white images at 300 dpi. Image quality is notably higher for documents belonging to the `Invoices-2` group. Documents of the other two groups have been obtained converting only the first page of the corresponding PDF, which was available on-line, to black-and-white images at 300 dpi. In about half of the cases, these PDFs were likely obtained by scanning the corresponding paper documents.

Some classes of `Invoices-1` consist of documents whose original paper size is smaller than the A4 page format: these invoices have been scanned as A4 pages and positioned in a variable way with respect to the scanner area, resulting in images in which the content position is variable.

Each document image has been converted to a set of blocks (see Sect. 3.1) using an OCR system.[1] The OCR system was instructed to deskew images, if needed, and configured to perform as best as possible. Yet, some errors were introduced by the OCR system: in many cases, they were caused by various scanning artifacts. OCR errors can be classified in segmentation errors and text-recognition errors: the former result in blocks containing different text elements among different documents of the same class; the latter result in textual contents—i.e., l values—which are quite different from what actually written on the paper. We noted that segmentation errors usually imply text-recognition errors (see Fig. 2); moreover, poor print quality—e.g., documents produced by dot matrix printers—usually caused text recognition errors. We did include in the dataset also the documents for which the OCR system introduced errors affecting the blocks that contained the values being searched: this choice allowed us to assess our approach effectiveness in a very realistic scenario, with respect to such OCR errors.

The text contained in our dataset documents is composed of printed English characters. The language is Italian for `Invoices-1` and `Invoices-2`, English for `Data-sheets` group and many documents of the `Patents` group and other languages for the other documents.

We defined three schemata: the schema for groups `Invoices-1` and `Invoices-2` contains 8 elements, the schema for the `Patents` group contains 11 elements, and the schema for the `Data-sheets` group contains 8 elements. Table 2 shows the elements composing the three schemata.

Then, we constructed the ground truth for all documents, i.e., an operator inspected visually each document and (i) (ii) searched all elements of the corresponding schema; (iii) for each element found, manually selected the corresponding true sequence. In all documents of our dataset, the true sequence turned out to be composed of 2 blocks at most. True sequences composed of two blocks were usually composed of a block corresponding to a label and a block corresponding to the actual element field: for example, Fig. 2a shows a block containing a date label and a block containing the actual date value.

While manually building the ground truth, we found that the following situations occur: (i) classes whose documents never contain a given element—e.g., a given producer of a given type of electronic component does not provide the `Storage Temp.` information; (ii) classes for which only some documents contain a given element—e.g., given a patent source, some patent obtained from that source has a `Representative` value while some other has not; (iii) classes whose documents may contain multiple occurrences for a given element—e.g., a given document may contain a value for `Total amount` in more than one page or even more than once on the same first page. Concerning case iii, we manually clustered documents based on the occurrence of the corresponding elements—e.g., all values for `Total amount` occurring on top of first page of documents of a given class were separated from `Total amount` values occurring on bottom of first page. We say that these values correspond to exactly one *sub-element*—e.g., `Total amount (1)`.

Finally, we formed several *test sets*. Each test set corresponds to exactly one rule and contains all and only documents that: (i) belong to the same given class and (ii) contain exactly one value for the given element or sub-element. We obtained 591 test sets with size ranging from 2 to 31 documents. Figure 3 shows the number of test sets vs. their size. Note that more than 50% (304 on 591) of the test sets contains 5 or less documents.

6.2 Results with varying training set size

We assessed our approach effectiveness with respect to the size m of the training set. To this end, we proceeded as follows

[1] An early version (0.2) of the OCRopus™ open source document analysis and OCR system.

Table 2 The elements composing the three schemata that we applied to our dataset

Invoices-*	Patents	Data-sheets
Date	Title	Model
Document N.	Applicant	Type
Issuer VATIN	Inventor	Case
Customer VATIN	Representative	Power Dissipation
Customer Name	Filing Date	Storage Temp.
Total Amount	Publication Date	Voltage
Net Amount	Application N.	Weight
Taxes Amount	Publication N.	Thermal Resist.
	Priority	
	Classification	
	Abstract 1st line	

The groups are showed in the table header

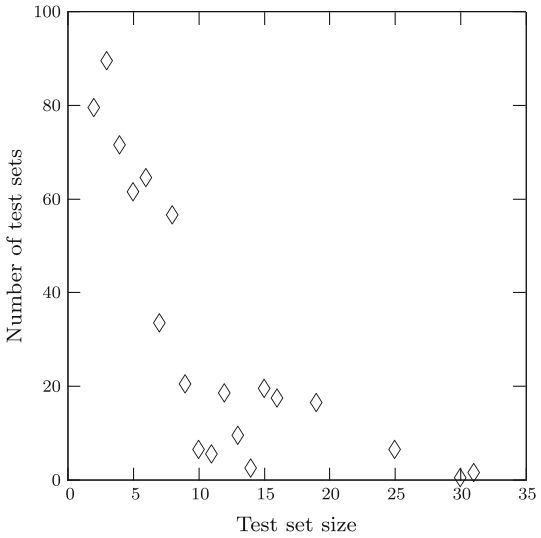

Fig. 3 Number of test sets with a given size

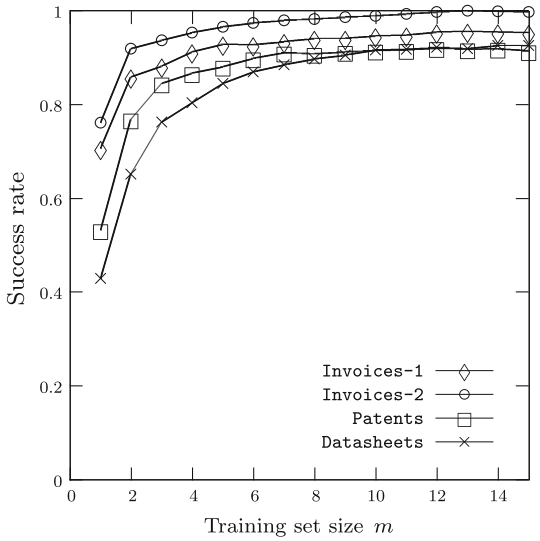

Fig. 4 Success rate versus training set size m for the four groups

for each test set: (i) we built the rule using a training set composed of m documents extracted from the test set; (ii) we processed each of the remaining documents of the test set and, for each document, we determined the block sequence found with the rule (i.e., the matching sequence). We considered a success the case in which the found block sequence corresponded with the document true sequence and a fail otherwise. The success rate is hence given by the number of successes divided by the sum of successes and fails.

In order to average the effect of possible outliers in the training set (e.g., documents for which printing or scanning quality was significantly lower than other documents of that test set), we repeated the experiment with up to 20 different training sets extracted at random from the test set. For example, with $m = 3$ and a test set of 4 documents, we performed 3 experiments, each one evaluating the remaining one document; with $m = 3$ and a test set of 10 documents, we performed 20 experiments, each one evaluating the remaining 7 documents. Only test sets with more than m documents concurred to the experimental evaluation of the success rate for training set size m.

Figure 4 shows results in terms of success rate vs. training set size m, one line for each different dataset group.

The main finding is that in all cases, the success rate tends to be greater than 0.90, with a perfect result (i.e., 1.0) for the `Invoices-2` group. Another important finding is that the proposed approach exhibits very good results even when trained with very few documents. For example, for `Invoices-2` group, just a single training document is sufficient in order to obtain a success rate greater than 0.75, while with $m = 2$, it increases to more than 0.90.

These results show also that the success rate is lower for `Data-sheets` and `Patents` groups. We verified that the rather low quality of the scanned documents of these two

groups caused a much larger number of OCR errors, both segmentation and text-recognition errors, which negatively affected the success rate.

6.3 On-line retraining

In this section, we analyze the possibility of on-line retraining of the system. By this, we mean an operating procedure in which document models are defined based on a very small training set and update whenever new documents of that model are processed, either automatically or with minimal involvement of the operator.

The rationale for this operating procedure is based on practical and important considerations. First, a small training set implies that the operator has to mark a smaller set of documents. Since the operator is involved for less time, the potential for cost savings is evident, in particular when the volume of documents and models is large. Second, although results in the previous section demonstrate that our approach is effective even with only a few samples, they also show that performance generally improve with the size of the training set. It follows that updating a document model when new samples become available is useful. Third, a large training set could not be available, or it could become available only after a long time. In these cases, it may be mandatory to start processing documents as soon as possible, even with slightly worse performance, and improve the system if and when new samples become available.

In order to address these considerations, we reasoned in term of retraining policies. A *retraining policy* defines whether and how an existing rule is affected by the elaboration of a new document of the given class. We considered three possible retraining policies. In all cases, we assume that the current training set D from which the rule was generated is available.

- "Unsupervised" A new training set D' is built adding each newly processed document to D and the rule is rebuilt on D'. In other words, the matching sequence is always considered to be the true sequence, irrespective of whether this is indeed the case.
- "Supervised" The operator is asked to confirm that the matching sequence corresponds to the true sequence and, if not, correct the sequence. Then, a new training set D' is built adding the newly processed document to D, and the rule is rebuilt on D'.
- "Hybrid (t)" This policy corresponds to the "Supervised" policy, if the size of the current training set D is lower than a given threshold t, or to the "Unsupervised" policy, otherwise.

Policies "Unsupervised" and "Supervised" require different amounts of involvement by the human operator. The first policy is fully automatic as it requires no intervention at all. The second policy requires the operator to either accept or reject, and possibly correct the processing result. In a practical implementation, this step could correspond to just few clicks, given that a visual representation of the matching sequence is provided to the operator. Policy "Hybrid ()" is similar to "Supervised", except that the involvement of the operator is no longer required after the training set size has reached the threshold.

We performed an experimental evaluation of the proposed policies as follows. We selected the 110 test sets (as defined in Sect. 6.1) with at least 10 documents. We chose to discard the other test sets, because their small size makes them not very meaningful for this specific experiment. For each of these 110 test sets, we generated 20 sequences using 20 set permutations, in order to average the effect of possible outliers in the sets. Then, for each sequence, we: (i) built a rule using a training set composed of *only one element*, corresponding to the first document of the sequence; (ii) for each subsequent document in the sequence, we fed it to our prototype and updated the rule according to the given policy (iii) and we proceeded with the next document.

Figure 5 shows results in terms of success rate vs. sequence index k ($k = 1$ for the first document in the sequence, 2 for the second, and so on), one line for each different retraining policy.

As expected, the "Supervised" policy delivers the best performance: a detection rate greater than 0.90 is reached after 7 documents have been evaluated. This result confirms what already found with the former experiments: recall that Fig. 5 shows the success rate averaged over all the 4 groups.

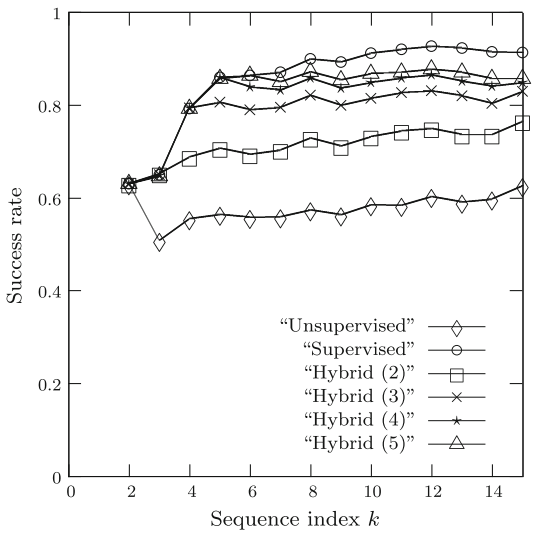

Fig. 5 Success rate versus sequence index for the three updating policies

With the "Hybrid (t)" policy, the success rate increases with k even when only 2 processing results have been confirmed by the operator. This is interesting because the training set potentially contains noisy elements.

The "Unsupervised" shows to be inadequate. We think the motivation of this result is in the rather low success rate that is obtained with $k = 2$ (i.e., with a training set size $m = 1$): this causes this policy update the training set with a new element which, in about 40% of the cases, is not the true sequence.

Clearly, the practical impact of the above findings will depend strongly on nature and scale of the system, in particular, on the amount of feedback from operators that may actually be used for the retraining policies.

For example, a small-scale system could be synchronous, i.e., any new job for a given class will not be submitted before the previous job result has been received by the user. In this case, the user would perceive a detection rate as showed in the best line of Fig. 5. A large scale system, on the other hand, could work asynchronously: users submit document-processing jobs and then extract and check their results sometime later. Several documents of a given class could be elaborated before a user feedback for a previous job of that class is available to the system—e.g., few users requiring batch elaborations, or several users. In this case, each user might hence perceive an actual detection rate that stays, at the beginning, between the worst and the best lines of Fig. 5. The key fact, however, is that the detection rate will ultimately tend to the greatest values found in our experimentation—i.e., about 0.95 on the average, and up to 1.0 for good quality documents—basing on the assumption the users' feedback is eventually provided to our system.

6.4 Performance

All the experiments described before have been executed on prototype written in Java 1.6SE, running on a quad core 2.50 Ghz PC with 4 GB of RAM. The average time for building a matching probability goes from less than 0.5 ms, for $m = 1$, to less than 3.5 ms, for $m = 20$, growing linearly with m. The average time for searching for a single element on a document, given its blocks representation, is about 15 ms. To place these figures in perspective, it suffices to note that an OCR execution on a single page document of our dataset takes about 12 s, on the same hardware.

7 Conclusions

We have proposed an approach for information extraction from multi-page printed documents. The system operates on documents represented in terms of text blocks, as obtained using an OCR system, each block simply consisting of position and size attributes and textual content. The approach is based on a probabilistic framework, not bound to a specific domain or type of documents and does not require any hand-coded knowledge. We derived a simple parametric form for the probability that a given block sequence contains the required information. Our contribution consists also of a procedure for building a model for a new document class, given few samples, and a procedure for extracting all the relevant information from a document, using the model. The form that we derive for the probability is general yet simple enough to allow us applying the maximum likelihood method and to do so by means of a class-independent GUI that does not require any specialized IT-related skills.

We evaluated the performance of our proposal on a dataset composed by 807 multi-page printed documents belonging to three different domains: invoices, patents and electronic components data-sheets. We also proposed and evaluated a method for automatically upgrading a model definition as more and more documents of the model are processed. This method would allow keeping human involvement to a minimum and would be a necessity whenever the arrival rate of documents belonging to the new model is very low. The experimental results show that our proposal is very effective, even when the training set is composed of very few samples.

Acknowledgments The authors are grateful to the anonymous reviewers for their detailed and helpful comments.

References

1. Aiello, M., Monz, C., Todoran, L.: Combining linguistic and spatial information for document analysis. Arxiv preprint cs/0009014 (2000)
2. Aiello, M., Monz, C., Todoran, L., Worring, M.: Document understanding for a broad class of documents. Int. J. Doc. Anal. Recognit. **5**(1), 1–16 (2002)
3. Amano, A., Asada, N., Mukunoki, M., Aoyama, M.: Table form document analysis based on the document structure grammar. Int. J. Doc. Anal. Recognit. **8**(2), 201–213 (2006)
4. Bartoli, A., Davanzo, G., Medvet, E., Sorio, E.: Improving features extraction for supervised invoice classification. In: Artificial Intelligence and Applications. ACTA Press (2010)
5. Belaid, Y., Belaid, A.: Morphological tagging approach in document analysis of invoices. In: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04), vol. 1–01, pp. 469–472. IEEE Computer Society (2004)
6. Van Beusekom, J., Keysers, D., Shafait, F., Breuel, T.M.: Distance measures for layout-based document image retrieval. In: Second International Conference on Document Image Analysis for Libraries, 2006. DIAL'06, p. 11 (2006)
7. Cesarini, F., Francesconi, E., Gori, M., Soda, G.: Analysis and understanding of multi-class invoices. Int. J. Doc. Anal. Recognit. **6**(2), 102–114 (2003)
8. Cesarini, F., Gori, M., Marinai, S., Soda, G.: INFORMys: a flexible invoice-like form-reader system. IEEE Trans. Pattern Anal. Mach. Intell. **20**(7), 730–745 (1998)
9. Chen, N., Blostein, D.: A survey of document image classification: problem statement, classifier architecture and performance evaluation. Int. J. Doc. Anal. Recognit. **10**(1), 1–16 (2007)

10. Dengel, A. R.: Making documents work: Challenges for document understanding. In: Proceedings of the Seventh International Conference on Document Analysis and Recognition, vol. 2, p. 1026. IEEE Computer Society (2003)
11. Hamza, Hatem, Belaïd Yolande, Belaïd, Abdel: Case-based reasoning for invoice analysis and recognition. In: Case-Based Reasoning Research and Development, pp. 404–418 (2007)
12. Hu, J., Kashi, R., Wilfong, G.: Comparison and classification of documents based on layout similarity. Inf. Retr. **2**(2–3), 227–243 (2000)
13. Klein, B., Agne, S., Dengel, A.: On benchmarking of invoice analysis systems. In: Document Analysis Systems VII, pp. 312–323 (2006)
14. Klink, S., Dengel, A., Kieninger, T.: Document structure analysis based on layout and textual features. In: Proceedings of International Workshop on Document Analysis Systems, DAS2000, pp. 99–111. Citeseer (2000)
15. Kwok, Thomas, Laredo, Jim, Maradugu, Sridhar: A web services integration to manage invoice identification, metadata extraction, storage and retrieval in a multi-tenancy SaaS application. In: Proceedings of the 2008 IEEE International Conference on e-Business Engineering, pages 359–366. IEEE Computer Society (2008)
16. Lewis, D., Agam, G., Argamon, S., Frieder, O., Grossman, D., Heard, J.: Building a test collection for complex document information processing. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 665–666, Seattle, Washington, USA. ACM (2006)
17. Peng, H., Long, F., Chi, Z.: Document image recognition based on template matching of component block projections. IEEE Trans. Pattern Anal. Mach. Intell. **25**(9), 1188–1192 (2003)
18. Sako, H., Seki, M., Furukawa, N., Ikeda, H., Imaizumi, A.: Form reading based on form-type identification and form-data recognition. In: Proceedings of the Seventh International Conference on Document Analysis and Recognition—vol. 2, p. 926. IEEE Computer Society (2003)
19. Schulz, F., Ebbecke, M., Gillmann, M., Adrian, B., Agne, S., Dengel, A.: Seizing the treasure: Transferring knowledge in invoice analysis. In: Proceedings of the 2009 10th International Conference on Document Analysis and Recognition—vol. 00, pp. 848–852. IEEE Computer Society (2009)
20. Sorio, E., Bartoli, A., Davanzo, G., Medvet, E.: Open world classification of printed invoices. In: DocEng 2010: Proceedings of the 10th ACM Symposium on Document Engineering, ACM, New York, NY, USA (2010)
21. Todoran, L., Aiello, M., Monz, C., Worring, M.: Logical structure detection for heterogeneous document classes. In: Proceedings of SPIE, vol. 4307, p. 99 (2000)