Intellix - End-User Trained Information Extraction for Document Archiving

Daniel Schuster, Klemens Muthmann,
Daniel Esser, Alexander Schill
Computer Networks Group
TU Dresden
01062 Dresden, Germany
daniel.schuster@tu-dresden.de

Michael Berger, Christoph Weidling,
Kamil Aliyev, Andreas Hofmeier
DocuWare GmbH
Therese-Giehse-Platz 2
82110 Germering, Germany
michael.berger@docuware.com

***Abstract*—Automatic information extraction from scanned business documents is especially valuable in the application domain of document archiving. But current systems for automated document processing still require a lot of configuration work that can only be done by experienced users or administrators. We present an approach for information extraction which purely builds on end-user provided training examples and intentionally omits efficient known extraction techniques like rule-based extraction that require intense training and/or information extraction expertise. Our evaluation on a large corpus of business documents shows competitive results of above 85% F1-measure on 10 commonly used fields like document type, sender, receiver and date. The system is deployed and used inside the commercial document management system DocuWare.**

I. Introduction

Information extraction as a core technology for automated document processing has been quite successful in a number of application domains. Commercial solutions like smartFix [1] and the Open Text Capture Center [2] automatically process scanned invoices, medical documents or insurances on a regular basis. Information included in the documents is extracted and documents are automatically aligned to existing business workflows as well as attached to existing ERP systems and other databases enabling a high accuracy of extraction results.

While this works well for large and medium-sized institutions, it still requires a high effort of on-site configuration and thus does not fit the needs of small office and home office (SOHO) users. To bring information extraction to SOHO users, our goal was to reduce configuration and adaptation efforts as effectively as possible.

Purely training-based approaches will never reach the accuracy and comprehensiveness of customized or specially adapted solutions. Nevertheless, they provide an added value for users who do not have the possibilities to run a customized solution. Every field the system is able to extract with high precision is of help for SOHO users while recall may still be low. This is discussed in more detail in Section II.

Our work describes an approach for purely training-based information extraction involving text and layout features. Its core technology is an identification of similar looking documents on top of a local search engine (Apache Lucene). The approach described in Section III learns immediately from user feedback, does not require any training documents in advance and works completely language-independent.

To underline the highly adaptive character of our approach, we test our solutions with what we call *Cold Start Metrics*, i.e., a gold standard evaluation starting with an empty learning model and adding each document not recognized correctly as a training example. Results are shown and discussed in Section IV.

II. Requirements and Related Work

We started Intellix as a publicly funded research project with the primary focus on document archiving for SOHO users. We believe that any solution in this market will only be successful if it significantly lowers the burden of manually tagging new documents. Otherwise, SOHO users will just stay with their current approach which is mainly a hierarchical folder structure often in combination with public cloud services such as Dropbox.

Thus the research task to be solved is nearly configuration-free information extraction or functional role labeling (compare [3]) of a few commonly used fields in document archiving. A minimal set to enable structured archiving consists of *document type*, *sender*, *receiver* and *date*. For more user comfort we added further popular fields (*contact number, amount, customer ID, subject, document number, to be paid*) based on a survey carried out by DocuWare.

Our goal is to be able to extract these 10 commonly used fields with an accuracy of above 80%. Furthermore, we want the algorithm to quickly adapt to new types of documents. Users expect the system to extract information from a document looking similar to an already tagged document immediately. We take this into account when we define our evaluation metrics in Section IV.

It is important to mention that we do not want to extract more specific information like full table content (see [1]) or handwritten information. Clearly, pure training-based approaches will hardly work for these more advanced information extraction tasks.

If we look at common solutions in the field of information extraction out of business documents, we can divide them into two different kinds of approaches.

Knowledge engineering systems extract information with handcrafted rules from target documents. Rules range from simple regular expressions [4] over the identification of relational key words [5] to more complex patterns for identifying addresses [6] or table contents [7]. For breaking up the strict matching of rules, Klink et al. [8] provide a more sophisticated solution. The authors allow combining multiple types of exchangeable rules, using a fuzzy-matching to check them. Although this kind of solutions produce excellent extraction results, they do not match our requirements for low configuration and high flexibility. Thus we do not use any kind of components requiring rules in our current implementation.

Self-learning or trainable systems avoid the high effort for manual engineering of extraction knowledge and fit much more to the needs of SOHO users. These systems use annotated example documents and try to automatically find extraction mechanisms like generated rules or statistical models. Cesarini et al. [9] identify similar documents within the training set and generate positional rules that can in turn be used for extraction. We have also presented a similar approach in the past [10]. Matsumoto et al. [11] also considers layout characteristics like bold or italic characters for rule generation. Medvet et al. and Bart et al. [12], [13] use a probabilistic model to detect index fields and table content. Pandey et al. [14] make use of similarity measures for detecting relevant information. A huge disadvantage of current trainable systems is the need for large sets of example documents, requring manual annotations. While this problem has not yet been deeply discussed in the information extraction domain, the influence of training on the performance of classifiers is well studied [15].

We do not know an information extraction system, neither handcrafted nor trainable, that allows the extraction of fields relevant for document archiving and performs as fast and adaptive as our approach. Especially in terms of zero configuration to set up and run the system, our approach is superior than state-of-the-art approaches.

III. Approach

Our approach, the Intellix indexing process, is shown in Figure 1. The process starts with a document in post OCR representation, i.e., an XML file with a hierarchical structure of the document beginning at the page level turning down to text and table zones, lines, words and characters. Each element carries position information so that bounding boxes of words can be derived from the XML file. Such an XML representation is the usual output of commercial OCR engines. We rely on external OCR because of its superior recognition rates. Hence, OCR optimizations are not part of our work.

A. Classification

Like most document processing systems, our approach is divided into classification and information extraction. Classification assigns a basic document type such as invoice, dunning, or delivery note to the input document. After experimenting with very diverse features for this purpose (simplified layout representation, logo detection, ...) we surprisingly found out that a simple Bag of Words model in combination with a kNN classifier showed best results on our dataset.

We thus extract and count all words in the input document. Lucene [16] is used as a kNN Classifier where each known training document is indexed and then ranked according to the words included in the input document. Lucene ranking mainly relies on the well-known TFIDF (term frequency / inverse document frequency) ranking scheme. The document type of the input document is then determined by majority voting on the ranked list of k retrieved similar documents. Currently k is always set to five.

Parallel to document type classification, we try to identify the document template. This is not just a subtype of the document type as there are some cases where a company uses the same template with little modifications for different document types. What we call template rather identifies a transformation function of some input data and fixed elements to a graphical document representation. The key idea of any document processing system is to reverse this transformation to identify the input data used to create the representation.

While it is quite easy for a human to identify documents using the same template, machines need sophisticated algorithms to make a decision. For our Template Detection component we tested different techniques based on graphical and layout features like the simplified document layout representation described in [17]. Again, a Lucene-based kNN approach was desirable for our SOHO use case as it enables fast detection and immediate learning. We found a solution called Wordpos Features. We generate a grid of rectangular areas, e.g. a 20 by 30 grid for A4 paper. Each word in the input document is assigned to the grid cell where its upper left corner originates.

These features perform exceptionally good for template identification. More details about this can be found in [10]. Again, with Lucene's built-in TFIDF-based ranking, we isolate the fixed words at invariant positions used in each template. Lucene returns a ranked list of k training documents best matching the template of the input document. Normalizing the Lucene score to the first entry in the list, a threshold can be used to filter out non-matching documents. The list of matching template documents is then handed over to the different indexer components.

B. Information Extraction

Each indexer component takes the input document and the list of matching template documents and returns extracted data and an extraction score for each candidate. We developed one indexer each for fields with fix value across template documents (Fix Field Indexer), fix position but variant value (Position-based Indexer) and variant position / variant value (Context-based Indexer). But more algorithms can be inserted easily in the extraction workflow.

The Fix Field Indexer compares tagged values per field in the known template documents. If there is high agreement, the according field of the input document is very likely to have this value, too. This often occurs for the field *sender* as most templates are exclusively used by one company only. But there are some rare cases where multiple companies use the same software to generate documents and thus may produce the same template. In this case, the fix field indexer will not return a value for the field *sender*.

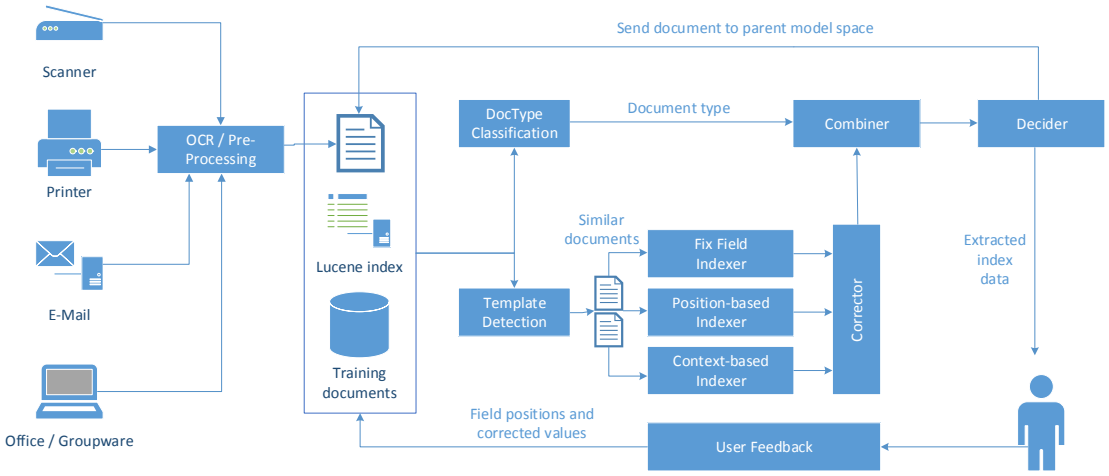

Fig. 1. Intellix indexing process

A large number of interesting fields like *sender, receiver, date, document number*, etc. are always placed at the same position within documents of the same template. The Position-based Indexer identifies these fields according to bounding box overlap of corresponding fields in the template documents. We use the Jaccard index [18] for scoring candidates. This performs well for most of cases but sometimes finds wrong fields perfectly matching the position and size of a bounding box in the input document. The Combiner component mitigates this undesirable effect later on.

For fields with invariant positions like total amount, we use context information of each candidate token to possibly assign it to a field used in the template documents. E.g., the text token "Total amount" or one of its synonyms often appears left to or top of the actual value of the field *total amount*. As the search space is small (we use a maximum of 5 to 10 template documents), this algorithm still performs quite fast although each template document and the input document have to be compared word by word with each other.

The Corrector component carries out typical OCR error corrections and other user-defined corrections. This is again completely based on end user feedback. Whenever the user corrects a field manually, this mapping is saved and later tried to apply to other extraction results based on error heuristics.

C. Result Aggregation

The Combiner component merges results of the kNN Classifier and the three Indexers mentioned above. Weights for each combination of field and indexer are applied to generate a ranked list of candidates per field. The candidate with the highest score for each field is selected. Details on the methodology can be found in [19] where we still worked with a combination of training-based and rule-based extractors. The same methodology is used to calculate the best weights for the combination of purely training-based algorithms as shown in Figure 1. We use a large document corpus of tagged business documents to find the best combination of field-based weights. This matrix is used for new instances of a user model space.

D. Hierarchical Extraction

The last step in our extraction process is a component called Decider. It collects schema information for each document type to know the fields that are commonly used by each type. For this purpose it analyzes the user feedback over time and thus is able to build a complete schema with no initial training phase required. Simply speaking, it knows the probability that a certain field is important for a user. If the cumulated extraction scores of these important fields are below a certain threshold, the Decider calls a second extraction process. We call this a parent model space where documents not recognized by local extraction processes are collected and provided for a number of end-user model spaces.

Technically, involving the parent model space means running the same process of Figure 1 for a second time. But this time with a larger number of training documents as provided by different child model spaces. If successful, extraction data (field positions and values) is provided back to the child process and may be added as training data. Documents are never sent from the parent to a child model space as they may contain sensitive information of other organizations.

E. User Feedback

Finally, the extraction results are presented to the user. He may either accept the results with one click or correct them with a graphical tool where the user marks the occurrence of the value in the document. This is then added as a training document to the local model space. Replacement strategies for training documents and more details about feedback handling in Intellix can be found in [20].

IV. Implementation and Results

We implemented the extraction process described above both in a prototype implementation at TU Dresden and an integrated product version at DocuWare. Figure 2 shows the end-user view of the extraction system. Newly scanned documents appear on the left hand side and are already indexed using the Intellix process. Green, yellow and red markers show the estimated accuracy of the extraction results. Users should control documents marked yellow and red and correct them

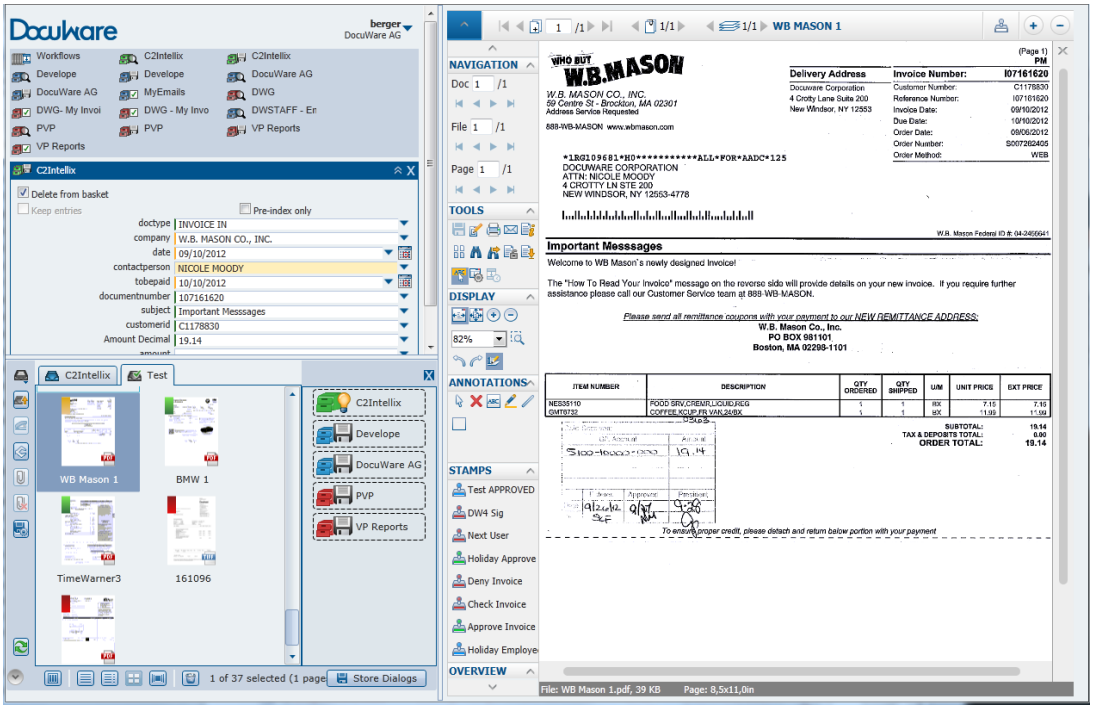

Fig. 2. Screenshot of Intellix indexing integrated in the DocuWare document management system

if necessary. For this purpose they either use drop down lists with proposed values or click at the information snippet in the graphical document representation to the right.

We did a number of evaluation runs to assess the extraction quality of the extraction process. For these runs we used random subsets from a set of approximately 12,000 manually annotated documents provided by DocuWare. Results are shown in Figure 3. We present numbers for micro and macro averaged precision, recall and F1-measure per field as well as combined scores according to Chinchor and Sundheim [21].

All tests in the column *Local* are done on empty model spaces using a test procedure we call *Cold Start Metrics*. Each document not recognized correctly by Intellix, is added as a training document to the model space thus improving future indexing. While the extraction rates averaged across 4,000 documents are quite good (above 85% F1-measure), this cold start approach clearly shows a bad performance in the starting phase as can be seen by the results in the F1 start column. This corresponds to the first-hand impressions we had when rolling out a first field test of the system with DocuWare customers. The training only approach requires at least one or two documents per template until extraction can work properly. Customers sometimes get frustrated in the starting phase and are not willing to continue tagging work.

This bad start experience can be improved in two ways: Rule-based extraction could be provided as an add-on to extract at least some fields that adhere to known patterns such as VAT ID. This minimal rule set can be managed centrally thus not requiring any on-site configuration for the SOHO user. The other (obviously better) approach is to profit from training done by other users in their model spaces. We tested the efficiency of the distributed approach by employing a single pre-trained parent model space with 8,000 documents. The child model space calls the parent whenever it gets no sufficient results. As can be seen in the column *Distributed F1*, this does not improve extraction results for the 4,000 document test significantly. But it really improves the start experience (*Distributed F1 start*) by 22 - 29%. Thus, our approach to sharing extraction knowledge between organizations is especially helpful in the starting phase of the system. It may later be deactivated. It is interesting to see that the F1-measure for the ToBePaid field actually decreases for the distributed case. The reason is that fields are not equally distributed over our dataset and there are many document not containing the ToBePaid field at all. So the parent model space outvoted its child because it assumes that there is no ToBePaid field and thus removed some correct extractions done by the child model space.

The average runtime per document for the whole Local process is 98.65 ms on a commodity PC starting with the post-OCR representation of the test document. This shows the speed advantage of the Lucene-based approach. The Distributed use case sends about 10% of the documents to the parent increasing average document processing time to 124.41 ms. While only 10% of documents are sent to the parent, processing time increases by more than 20% which is partly due to the higher number of documents in the parent. This shows the need for our hierarchical approach. It would be far less efficient to process all documents in one common cloud model space.

While we were not able to run a comparative test with other systems on our document set, results in Table 3 show astonishingly good and competitive results provided that our approach requires no training or on-site configuration in advance. The approach is completely language-independent and may be used for any kind of document type. New fields may be defined at any time by the user. New document templates only require manual tagging a few times or even only once.

	Local				Distributed		Improvement	
	Precision	Recall	F1	F1 start	F1	F1 start	F1	F1 start
Total (micro averaging)	0.877	0.863	0.870	0.707	0.894	0.864	3%	22%
Total (macro averaging)	0.863	0.855	0.857	0.667	0.889	0.858	4%	29%
DocType	0.966	0.965	0.966	0.925	0.968	0.941	0%	2%
ContactNumber	0.894	0.875	0.884	0.768	0.909	0.930	3%	21%
Amount	0.866	0.785	0.824	0.606	0.848	0.771	3%	27%
Sender	0.748	0.753	0.751	0.515	0.785	0.723	5%	40%
Date	0.942	0.913	0.927	0.740	0.961	0.970	4%	31%
CustomerId	0.941	0.919	0.930	0.713	0.956	0.957	3%	34%
Subject	0.923	0.912	0.917	0.721	0.947	0.912	3%	26%
DocumentNumber	0.883	0.886	0.885	0.774	0.935	0.849	6%	10%
ToBePaid	0.914	0.952	0.933	0.796	0.918	0.960	-2%	21%
Recipient	0.712	0.695	0.703	0.513	0.726	0.646	3%	26%
Runtime per document (ms)	98.65				124.41			

Fig. 3. Evaluation results (4000 test documents, 100 test documents for F1 start, 0 training documents for Local, 8000 training documents for Distributed)

V. Conclusion

We presented an approach for purely end-user training based extraction of index data from business documents. It is based on fast and accurate identification of document templates based on Wordpos features and Lucene. As shown in the evaluation we reach competitive extraction rates above 85% on 10 commonly used fields in document archiving.

Nevertheless, some parameters like the weights for result aggregation still have to be acquired on gold standard tests in advance. The ultimate goal of our work is competitive zero-configuration information extraction in the document processing domain. Our next steps to reach this goal are methods to automatically adjust combination parameters during runtime thus eliminating the need for fixed weights. In the long term, we try to leverage more potential out of cooperative information extraction. Model space hierarchies may consist of more than two levels including domain-specific model spaces or split model spaces for load balancing purposes.

Acknowledgment

This research and project was partially funded by the German Federal Ministry of Education and Research (BMBF) within the Framework Concept "KMU Innovativ" (fund number 01IS10011).

References

[1] F. Deckert, B. Seidler, M. Ebbecke, and M. Gillmann, "Table content understanding in smartfix," in *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, 2011.

[2] Opentext, "Opentext capture center," http://www.opentext.de/3/global/products/products-capture-and-imaging/products-opentext-capture-center.htm, 2012.

[3] E. Saund, "Scientific challenges underlying production document processing," in *Document Recognition and Retrieval XVIII (DRR)*, San Francisco, CA, USA, 2011.

[4] S. Adali, A. C. Sonmez, and M. Gokturk, "An integrated architecture for processing business documents in turkish," in *Proceedings of the 10th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing '09)*, 2009.

[5] B. Janssen, E. Saund, E. Bier, P. Wall, and M. A. Sprague, "Receipts2go: the big world of small documents," in *Proceedings of the 2012 ACM symposium on Document engineering (DocEng '12)*, 2012.

[6] B. Klein, S. Agne, and A. Dengel, "Results of a study on invoice-reading systems in germany." in *Document Analysis Systems*, 2004.

[7] Y. Belaid and A. Belaid, "Morphological tagging approach in document analysis of invoices," in *Proceedings of the 17th International Conference on Pattern Recognition (ICPR'04)*, 2004.

[8] S. Klink, A. Dengel, and T. Kieninger, "Document structure analysis based on layout and textual features," in *Int. Workshop on Document Analysis Systems*, 2000.

[9] F. Cesarini, E. Francesconi, M. Gori, and G. Soda, "Analysis and understanding of multi-class invoices," *IJDAR*, vol. 6, no. 2, pp. 102–114, 2003.

[10] D. Esser, D. Schuster, K. Muthmann, M. Berger, and A. Schill, "Automatic indexing of scanned documents - a layout-based approach," in *Document Recognition and Retrieval XIX (DRR)*, San Francisco, CA, USA, 2012.

[11] T. Matsumoto, M. Oba, and T. Onoyama, "Sample-based collection and adjustment algorithm for metadata extraction parameter of flexible format document," in *Proceedings of the 10th international conference on Artifical intelligence and soft computing (ICAISC'10)*, 2010.

[12] E. Medvet, A. Bartoli, and G. Davanzo, "A probabilistic approach to printed document understanding," *Int. J. Doc. Anal. Recognit.*, vol. 14, no. 4, pp. 335–347, 2011.

[13] E. Bart, "Parsing tables by probabilistic modeling of perceptual cues," in *Document Analysis Systems (DAS), 2012 10th IAPR International Workshop on*, 2012.

[14] G. Pandey and R. Daga, "On extracting structured knowledge from unstructured business documents," in *Proc IJCAI Workshop on Analytics for Noisy Unstructured Text Data*, 2007.

[15] C. Salperwyck and V. Lemaire, "Learning with few examples: An empirical study on leading classifiers," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*, 2011.

[16] The Apache Software Foundation, "Apache Lucene," http://lucene.apache.org, 2013.

[17] J. Hu, R. Kashi, and G. Wilfong, "Comparison and classification of documents based on layout similarity," *Information Retrieval*, vol. 2, no. 2, pp. 227–243, 2000.

[18] P. Jaccard, "Étude comparative de la distribution florale dans une portion des alpes et des jura," *Bulletin del la Société Vaudoise des Sciences Naturelles*, vol. 37, pp. 547–579, 1901.

[19] D. Schuster, M. Hanke, K. Muthmann, and D. Esser, "Rule-based vs. training-based extraction of index terms from business documents - how to combine the results," in *Document Recognition and Retrieval XX (DRR)*, San Francisco, CA, USA, 2013.

[20] M. Hanke, K. Muthmann, D. Schuster, A. Schill, K. Aliyev, and M. Berger, "Continuous user feedback learning for data capture from business documents," in *Workshop on Nonstationary Models of Pattern Recognition and Classifier Combinations under the framwork of HAIS2012*, Salamanca, Spain, 2012.

[21] N. Chinchor and B. Sundheim, "Muc-5 evaluation metrics," in *Proceedings of the 5th conference on Message understanding*, ser. MUC5 '93, Stroudsburg, PA, USA, 1993, pp. 69–78.