Rule-based vs. Training-based Extraction of Index Terms from Business Documents - How to Combine the Results

Daniel Schuster, Marcel Hanke, Klemens Muthmann, and Daniel Esser

TU Dresden, Computer Networks Group, 01062 Dresden, Germany

ABSTRACT

Current systems for automatic extraction of index terms from business documents either take a rule-based or training-based approach. As both approaches have their advantages and disadvantages it seems natural to combine both methods to get the best of both worlds. We present a combination method with the steps selection, normalization, and combination based on comparable scores produced during extraction. Furthermore, novel evaluation metrics are developed to support the assessment of each step in an existing extraction system. Our methods were evaluated on an example extraction system with three individual extractors and a corpus of 12,000 scanned business documents.

Keywords: Information Extraction, Result Combination, Document Management, Production Document Processing

1. INTRODUCTION

While the paperless office has been propagated since years, organisations world-wide still exchange huge amounts of paper documents each day. Especially in large organisations these documents are scanned, OCR-ed and stored in a doument management system, where index terms like *sender*, *receiver*, *date*, or *amount* are needed to be able to later find and group these documents. Automatic extraction of such index data is thus a crucial functionality of modern document management and document archiving solutions.

Current solutions for automatic extraction either use rule-based extraction or training-based extraction. Rule-based extraction works on a large set of hand-crafted rules like left and right context words of index values or regular expressions, e.g., matching VAT identification number. Training-based solutions solely depend on training examples given by the end-user and try to figure out certain patterns in the training documents. These may be based on similar page layout, characteristic words, logos or other graphical elements.

Rule-based approaches are best suited for use cases where large volumes of similar documents have to be processed like in hospitals or big companies. On the other hand, training-based approaches are flexible to adapt to changing document types and thus are preferred to be used in small office/home office (SOHO) scenarios. But these training-based approaches offer low performance in the starting phase where there are only few tagged documents in the training set. Thus, a combination with rule-based extractors would be desirable to help out with the deficits of training-based extraction especially in (but not limited to) the starting phase.

This work tackles the problem of how to combine results from multiple extractors for automatic processing of business documents. The research problem might look quite simple in the first place. If each extractor decides for one best-matching candidate for each field (*sender*, *receiver*, *date*, ...), we could combine the results by majority voting. But majority voting does not work if we have an equal number of votes for two extraction candidates. Certainly, extractors should rather return a ranked list of candidates with a score for each candidate. But this opens a new challenge: How do we ensure that scores from all extractors are comparable? This problem has not been considered yet in the literature as only similar classificators have been combined. We deal with the problem of how to combine results of an extractor working on a hand-crafted rule set with the results from training-based

Further author information:
Daniel Schuster: daniel.schuster@tu-dresden.de, Marcel Hanke: marcel.hanke@tu-dresden.de, Klemens Muthmann: klemens.muthmann@tu-dresden.de, Daniel Esser: daniel.esser@tu-dresden.de

extractors. In this case, normalization of extraction scores is an essential prerequisite for the combination of results.

The main contribution of this work is a process for the combination of extraction algorithms. It is designed to work well in the scenario described above but is not limited to extraction of index data from business documents. Besides the description of the process and its steps, we provide new evaluation metrics for the assessment of each of these steps during the process. Both contributions are rather guidelines than ready-to-use algorithms. They need to be adapted to the concrete combination problem in question. But the examples and evaluation results given in this paper clarify the process and hopefully make it easy to use it in other settings as well.

2. RELATED WORK

Existing approaches for extraction of index data from business documents or Functional Role Labeling[1] can be categorized in Text-based Extraction, Rule-based Extraction, and Layout-based Extraction (see Figure 1).

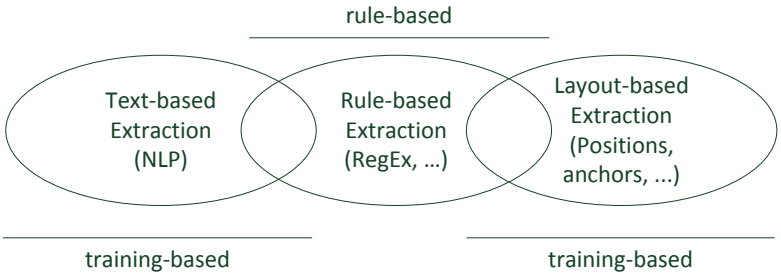

Figure 1: Existing approaches for extraction of index data from business documents

Text-based Extraction involves methods from Natural Language Processing (NLP) such as Tokenization, Stemming, and Part-of-Speech Tagging to assess the likelihood of a text token to belong to one of the predefined fields. The extractor itself is then often realized as a machine learning algorithm like Naive Bayes building a multi-class classification model on the basis of a set of training documents.[2] This approach is not only used for production document processing but also in other domains like medicine.[3] As can be seen in Figure 1 there is some overlap with rule-based approaches as features for these machine learning algorithms do not need to be only words or N-grams. Higher-level features like the match of a certain regular expression can be integrated to gain higher classification accuracy. These higher-level features need to be specified by a domain expert and are thus quite similar to hand-crafted rules.

This leads to the second class of extractors which work on a pre-defined rule set as it is common industry practice. These rules are often very domain-specific and dependent on language, culture, and other characteristics of the documents available for the rule engineers. A huge human effort is required to create and maintain an organization-specific rule set. This work certainly pays off for large organizations as highly optimized rule-based extractors offer higher accuracy compared to other approaches. Rule engines with regular expressions often require huge amounts of processing time which can be reduced using finite state machines.[4]

The third group of approaches is based on the layout of business documents. These documents are typically created using some template and one can find a lot of similar-looking documents in any large document corpus. Hu et al.[5] as well as Esser at al.[6] describe approaches to identify documents with the same template out of a document set either on a graphical or post-OCR representation of newly scanned documents. Once one or more documents of the same template are found, index data can be extracted based on known absolute positions or positions relative to anchor points in the document. Other works try to identify $(key, value)$ pairs out of OCR information[7] or even raw document images.[8] There is again some overlap with rule-based approaches as regular expressions can help to label tokens as date, ID, e-mail address or other and thus avoid extraction errors when only comparing the positions.

Hanke et al.[9] show that layout-based approaches can even be used in a cold-start scenario where there is no training data available in the beginning. Users give feedback on fields not recognized thus building up a training set which constantly improves the extraction results. After less than 10% of the corpus used for evaluation, the

system already reaches its long-term quality level. But the experience in the starting phase is rather bad which calls for rule-based methods to be used in conjunction with the training-based approach. Our assumption is that there is often a need for both, training-based as well as rule-based extractors in Functional Role Labeling. Thus we need methods to combine the results.

Early work on combination of different classification algorithms dates back to Kittler et al.[10] In this work, emphasis is already put on classifier selection, as classifiers should be selected carefully to decide if the additional processing time pays off. Ideally, there would be disjunct results of the different classifiers. If there is only one classifier available, variation can be done regarding the feature sets used by the classifier.[11–13] A separate feature set with the same machine learning algorithm would account for another single extractor in our understanding.

Combination methods discussed in these works are majority voting, Borda selection, maximum score selection, mean score selection, sum of scores selection, and product of scores selection. More sophisticated combination methods[14,15] or selection methods[16] are based on the Dempster-Shafer theory (DST) of evidence.[17] While DST allows for weighting different single extractors, it does not differentiate between different tasks or fields where algorithms may differ in performance. DST results of Giacinto et al.[16] only show marginal improvement compared to simple majority voting.

A combination of different approaches in Natural Language Processing is shown by Srihari et al.[18,19] Although the authors join rule-based and training-based techniques to extract Named Entities, they do not combine the extraction results from each algorithm by applying a score or weighting. They rather use different techniques to get an union set of distinct entities not evaluating them according to which or how many algorithms had extracted.

All these existing approaches combine machine learning classifiers which deliver either none or accurate and comparable extraction scores in their decision. Our approach is different from these approaches, as we combine results from training-based extraction with results from rule-based extraction. This requires adjusting available scores as described in Section 6. Furthermore we propose a fine-granular weighting scheme which takes the strengths and weaknesses of single extractors on different fields into account. Before discussing our approach in more detail, we will start with a precise definition of the research problem.

3. PROBLEM DEFINITION

The problem of combining results from multiple single extractors in Functional Role Labeling can be formalized as follows:

Given is a set of single extractors E where each extractor E_j takes the same test document d_i from the set of all test documents D as input and extracts results. For each document d_i, each single extractor E_j returns a set $R_{i,j} = \{r_{i,j,1}, .., r_{i,j,n}\}$ of results $(E_j(d_i) = R_{i,j})$. Each result $r_{i,j,k} = (field_k, \{(value_1, score_1), ..., (value_x, score_x)\})$ is a pair of a field $field_k$ and a set of extracted candidates for this field. The set F defines all fields to be extracted by the system. A candidate set consists of $(value, score)$-pairs with $score$ in the range [0..1]. Finally, the combiner is a function which takes a document-specific set of result sets $R_i = \{R_{i,1}, R_{i,2}, ..., R_{i,n}\}$ and returns a combined result set $R_{i,C}$.

This first part of the definition just defined the formats of the input and output of the single extractors and combiner and should be clarified with the following example:

$$
\begin{aligned}
F &= \{sender, receiver, date\} \\
D &= (d_1, d_2, d_3) \\
E &= (E_1, E_2) \\
R_{1,1} &= \{(sender, \{(TUD, 0.9), (DocuWare, 0.7)\}), (date, \{(4/12/2012, 0.7)\})\} \\
R_{1,2} &= \{(sender, \{(DocuWare, 0.8)\}), (receiver, \{(TUD, 0.8)\}), (date, \{(5/20/2012, 0.8), (4/12/2012, 0.5)\})\} \\
R_{1,C} &= \{(sender, \{(DocuWare, 0.75), (TUD, 0.45)\}), (receiver, \{(TUD, 0.4)\}), (date, \{(4/12/2012, 0.6), (5/20/2012, 0.4)\})\} \\
R_{2,1} &= ... \\
&...
\end{aligned}
$$

Now we are able to select the best matching candidates (maximum score) from each result set in order to find the subset of labels $L_{i,j}$ out of the set of all labels L that a single extractor assigns to a document d_i. The same procedure is done to identify $L_{i,C}$ as the set of labels assigned by the combiner for document d_i. A label is defined as a pair $(field, value)$. To assess the results of the extraction and combination, we need a ground truth consisting of a set of tagged test examples. For each document d_i from D there exists an example $L_{i,E}$ with a subset of labels from the set L of all labels.

The use of these symbols will be clarified again by an example:

$$
\begin{aligned}
L &= \{(sender, DocuWare), (sender, TUD), (receiver, TUD), (date, 4/12/2012), (date, 5/20/2012)\} \\
L_{1,E} &= \{(sender, DocuWare), (receiver, TUD), (date, 5/20/2012)\} \\
L_{1,1} &= \{(sender, TUD), (date, 4/12/2012)\} \\
L_{1,2} &= \{(sender, DocuWare), (receiver, TUD), (date, 5/20/2012)\} \\
L_{1,C} &= combine(\{L_{1,1}, L_{1,2}\}) = \{(sender, DocuWare), (receiver, TUD), (date, 4/12/2012)\}
\end{aligned}
$$

The ground truth example $L_{1,E}$ for document d_1 in the example above was much like we expected it to be except the date field. Here the combination algorithm returned the wrong result. Obviously, our goal is to minimize these errors or more precisely, at least do less errors than the best one of our single extractors. If this was not the case, we could simply select the best single extractor and only run this one on all test documents. We take the commonly used measures of Precision, Recall and F1-measure as our optimization targets. These measures are calculated for the combiner as follows:

$$Precision_{Combiner} = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|L_{i,C} \cap L_{i,E}|}{|L_{i,C}|} \tag{1}$$

$$Recall_{Combiner} = \frac{1}{|D|} \sum_{i=1}^{|D|} \frac{|L_{i,C} \cap L_{i,E}|}{|L_{i,E}|} \tag{2}$$

$$F1_{Combiner} = 2 \cdot \frac{Precision_{Combiner} \cdot Recall_{Combiner}}{Precision_{Combiner} + Recall_{Combiner}} \tag{3}$$

By replacing $L_{i,C}$ with $L_{i,j}$, the same method can be used to calculate $Precision_{Single}(j)$, $Recall_{Single}(j)$ and $F1_{Single}(j)$ of one single extractor. We are now able to formally define the goal of our combination approach:

$$F1_{Combiner} \geq argmax_{j=1}^{|E|} F1_{Single}(j) \tag{4}$$

Simply put, this is all we want to achieve. We can not expect both $Precision_{Combiner}$ and $Recall_{Combiner}$ to be always higher than what single extractors can achieve. It is possible to tune one single extractor for high precision and low recall and a second one vice versa. Thus, the F1-measure seems to be the best compromise. All other requirements like normalization of extraction scores rather emerged out of analysis of our single extractors as well as first experiments with combination which all led to worse results than the best single extractor thus violating Equation (4). These observations are the basis of our approach as described in the next section.

4. APPROACH

Our approach for the combination of single extractors in Functional Role Labeling is depicted in Figure 2. The steps to be carried out are selection of single extractors based on coverage results, normalization of extraction scores of the selected single extractors and finally the process of combination where one task is finding appropriate weights for the single extractors.

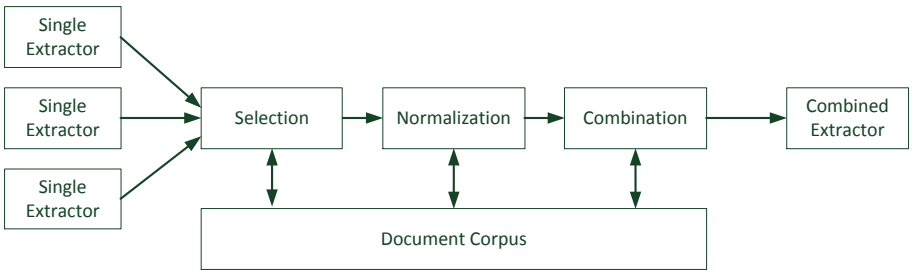

Figure 2: Combination approach

All steps have to be carried out on the basis of a document corpus, i.e., the set D and all tagged label sets $L_{i,E}$ as described in the problem definition. Results heavily depend on the size and diversity of the document corpus. As a consequence from the literature research, there will not be much effort to create sophisticated combination methods itself. We later rely on a simple weighted sum of scores. What matters more is the pre-selection and normalization of single extractors which will be covered in the next two sections.

5. SELECTION

As already pointed out by Kittler et al.,[10] pre-selection of single extractors helps in not wasting any resources in the extraction process. We propose the selection to consist of two steps: Pre-selection Evaluation and Iterative Coverage Test. In the first step, the **Pre-selection Evaluation**, single extractors will be judged on $Precision_{Single}(j)$, $Recall_{Single}(j)$ and $F1_{Single}(j)$ as described in Section 3.

In our case, we had three single extractors E_1, E_2, and E_3 which represent a text-based extractor, a rule-based extractor and a layout-based extractor. The document set D consisted of 12,494 business documents provided by our industry partner comprising 7 document types. The set of fields is defined as $F = \{$*sender, contactNumber, amount, customerID, email, documentNumber, subject, date, recipient, contactPerson, toBePaid*$\}$. All documents are fully tagged with examples.

Table 1 shows the results of the selection evaluation. The numbers should not be taken as a measure on how the three methods compare to each other in general. Our development effort was very much centered on layout-based extraction (E_3) while the other two methods were just implemented using state-of-the-art approaches. Especially the rule set of E_2 is not very much optimized. Its performance could be improved dramatically by more rule engineering effort.

Table 1: Pre-selection evaluation of the three single extractors

	E_1 (Text-based)	E_2 (Rule-based)	E_3 (Layout-based)
$Precision_{Single}(j)$	0.165	0.645	0.820
$Recall_{Single}(j)$	0.154	0.534	0.813
$F1_{Single}(j)$	0.159	0.584	0.816
Best field Precision	1.000 (document number)	1.000 (to be paid)	1.000 (contact person)
Best field Recall	0.614 (subject)	0.682 (contact number)	0.937 (customer ID)
Running time	283.1 ms	97.58 ms	37.92 ms

Based on this evaluation, there seems to be good potential for combination of the approaches. While E_3 performs best in all measures, E_2 still offers quite good results. E_1 is not competitive compared to the other two extractors. But for the selection step one should not rely only on the total number of correctly classified labels. As can be seen in the table, each extractor has some special fields where it performs quite good. If the user has a high interest in these fields, there might be a good reason to keep all three extractors.

To finally make the decision, we propose a method called **Iterative Coverage Test**. We initialize a working set of single extractor candidates with all single extractors from E: $E_{Cand} = E$. Furthermore, we create an empty set $E_{IC} = \{\}$ of single extractors which are to be combined in a perfect way, representing the ideal combiner (IC).

In each iteration step, we take the best performing single extractor from E_{Cand} and move it to E_{IC}. We then compare it to the remaining best extractor in E_{Cand}. We measure the total number of labels either recognized exclusively correct by the candidate extractor, non-exclusively correct, or wrong. The iterations continue until no single extractor is left in E_{Cand}.

In our case, we selected E_3 as the best extractor based on the results in Table 1. In iteration 1, E_2 was compared to E_3. In iteration 2, E_1 was compared to the ideal combination of E_3 and E_2. Table 2 shows the results of the Iterative Coverage Test for our use case.

Table 2: Results of Iterative Coverage Test

	Gain	*Coverage*	*Error*
Iteration 1: Add $E_j = E_2$ to $E_{IC} = \{E3\}$	**10.7**%	44.8%	44.5%
Iteration 2: Add $E_j = E_1$ to $E_{IC} = \{E2, E3\}$	**0.3**%	15.0%	84.7%

The columns of Table 2 show the share of labels assigned to the classes *Gain*, *Coverage*, and *Error*. *Gain* refers to all labels exclusively correctly identified by E_j, i.e., the true positives identified by E_j minus the true positives identified by E_{IC}. *Coverage* thus identifies the intersection of the sets of true positives identified by E_j and E_{IC}. Finally, *Error* subsumes all errors done by E_j, false positives as well as false negatives. This means we do not make a distinction between precision and recall in the Iterative Coverage Test, but rely on accuracy for simplicity reasons.

Now we are able to judge the potential of the combination. If we combine the two best extractors E_3 and E_2 we can expect a maximum gain of 10.7% in accuracy compared to running E_3 alone. If we then add E_1 we can only expect 0.3% of gain in correctly detected labels. Thus based on these results we select E_2 and E_3 and drop E_1 in the selection step.

The algorithm can be modified to run multiple times. We then do not select the best remaining extractor in each iteration but shift the order of candidate extractors to cover each possible arrangement of extractors in E. This is especially recommended if the Pre-selection Evaluation shows no big differences between the performance of the extractors.

6. NORMALIZATION

After selecting the best single extractors for combination, normalization of extraction scores has to be carried out. This is an essential step to ensure decidability as well as comparability. To judge these two properties of an extractor, we propose the **Extraction Score Distribution** as an evaluation measure. For each extractor E_j from E, each document d_i from D has to be processed by E_j to create the result sets $R_{i,j}$. For each field in F we now compare the result with the best score with the corresponding label from the ground truth example $L_{i,E}$. We count all correct and erroneous results and tag them with their score. Now we can create extraction score distribution diagrams based on score ranges. We propose to use the 10 bins $\{0.1, 0.2, ..., 1.0\}$ for the histogram.

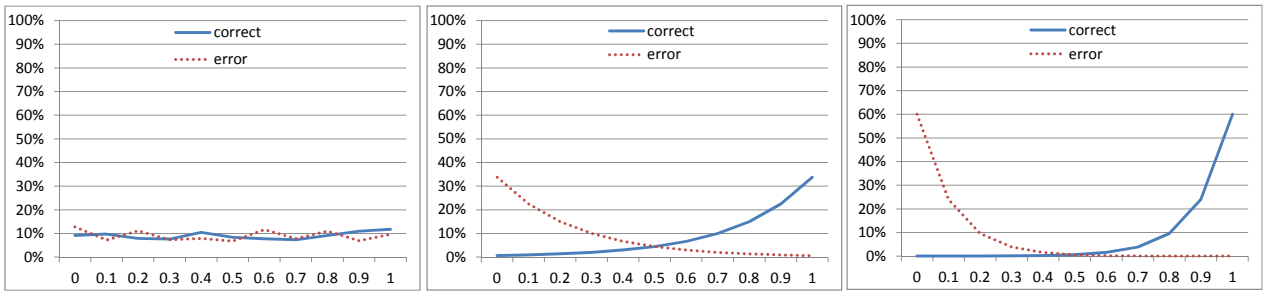

Figure 3: Patterns of extraction score distribution - left: worst case, middle: desirable case, right: best case

Figure 3 illustrates different hypothetical results. The left diagram shows a worst case distribution where it is not decidable if a result is an error or correct. If we draw a threshold line anywhere in the left diagram, we will always have roughly as much false positives as true positives. The diagram on the right is the best case we could imagine. We can draw a threshold line at $score = 0.5$ and are able to perfectly decide whether a value is correct or not. Unfortunately, this is an unrealistic case. What we rather expect is a distribution like the one in the middle of Figure 3. Here we can draw a threshold line which minimizes the amount of false positives and false negatives. With growing score, the possibility that a result is a true positive rises.

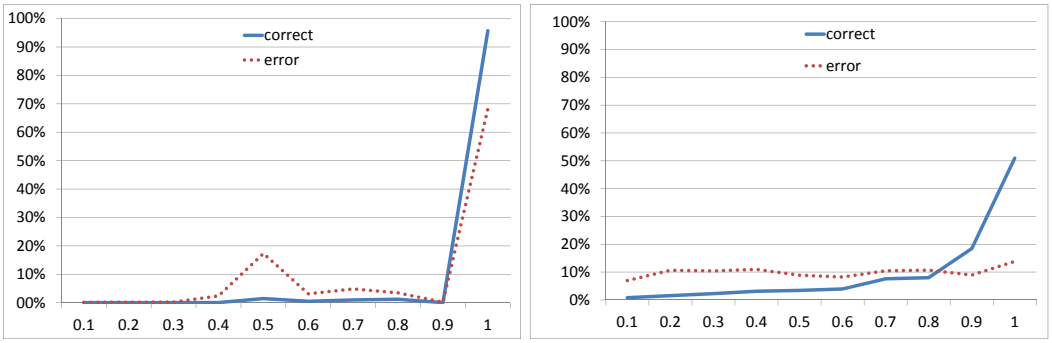

Figure 4: Extraction scores of layout-based extractor E_3 before (left) and after (right) normalization

For our use case we especially had to normalize the layout-based extractor E_3 as can be seen in Figure 4. On the left side of Figure 4 is the extraction score distribution of E_3 before the normalization step. There are peaks at score levels 0.5 and 1.0 where errors often occur but get a high score.

Extractor E_3 decides for best-matching document templates based on their Lucene score (see Esser at al.[6] for details). Template matches are determined by words and their positions in the training documents. Lucene was chosen because of its good performance as a kNN classifier. If a wrong template is selected as best match, we are not able to reject it, as Lucene scores are not normalized. Such wrong templates are normally rejected in the next step, as there are no values at the positions proposed by the wrong template. But in this step it may accidently happen to be some values at positions of the wrong selected template. Thus E_3 extracts these erroneous values and gives them a high score based on the template selection step. The left diagram of Figure 4 revealed that this happens more often than we expected.

We thus removed the template detection from the score calculation. We rather now only calculate the score based on the coverage of the extraction rectangle and the rectangle of the value found in the test document. Thus if we have a wrong template, we expect the positions to not fully match our test document's positions. This simple modification of the core algorithm not only improved the total performance of E_3 by nearly 6% but also smoothed the distribution of scores as can be seen in Figure 4 on the right.

We did similar steps in adapting E_2, especially replacing hand-wired weights for single rules by a probabilistic learning function for those weights. We do not want to go more into the details of our example normalizations as they are rather specific for each extractor under study. No matter what steps have to be taken to normalize extractor scores, the evaluation metric of Extraction Score Distribution should always be used before actually combining single extractors.

7. COMBINATION

After selection and normalization of the extractors, they can finally be combined using one of the combination approaches discussed in Section 2. We decided to use a weighted sum where weights are given for each pair $(extractor, field)$. We allow the user to specify non-negative integers as weights, as this is more intuitive. Each weight not specified is assumed to be 1. We end up with a configuration file like the one shown in Example 1. The weights are then normalized, to add up to 1 for each field:

Example 1 XML configuration file defining weights for the combination of E_3 and E_2.

```
<algorithm name="Layout Extractor">
  <weight field="amount" weight="1"/>
  <weight field="contactperson" weight="2"/>
  <weight field="customerid" weight="1"/>
</alogrithm>
<algorithm name="Rule Extractor">
  <weight field="amount" weight="4"/>
  <weight field="contactperson" weight="1"/>
  <weight field="customerid" weight="2"/>
</algorithm>
```

$$\forall F_k \in F : \sum_{j=1}^{|E|} weight_{E_j, field_k} = 1 \tag{5}$$

Now we calculate the combined score $Combscore$ as the weighted sum of scores for any distinct $(field, value)$ pair:

$$Combscore(d_i, field_k, value_x) = \sum_{j=1}^{|E|} weight_{E_j, field_k} \cdot score_{r_{i,j,k}} \tag{6}$$
$$field_k \in F, d_i \in D$$

The $(field, value)$ pair with the highest $Combscore$ is added to $L_{i,C}$, the set of labels selected for document d_i by the combiner.

There still remains one problem with this approach: Our problem definition allows the extractors to return an empty result for a field if they are unsure about its value. That's why we track precision and recall instead of accuracy. We want to value extractors that are built reluctant and better do return no value if there is a high chance to return an error. Table 3 shows an example where E_3 only extracted one value for a field while E_1 and E_2 returned 2 candidates with different scores.

Table 3: Combination example with an empty value

	E_1	E_2	E_3
weight	0.5	0.3	0.2
DocuWare	0.8	0.5	
TUD	0.5	0.6	0.8

Now we would calculate the combined score $Combscore$ like this:

$$Combscore(d_i, field_k, "TUD") = 0.5 \cdot 0.5 + 0.3 \cdot 0.6 + 0.2 \cdot 0.8 = 0.59$$
$$Combscore(d_i, field_k, "DocuWare") = 0.5 \cdot 0.8 + 0.3 \cdot 0.5 = 0.55$$

The missing value of E_3 is treated as if it had $score = 0$. Alternatively, we could re-normalize the remaining weights as some extractors may have been designed to only extract a few fields from F with high precision. In this case it would be more fair to calculate the combined score for the value "DocuWare" like this:

$$CombscoreLift(d_i, field_k, \text{"}DocuWare\text{"}) = \frac{1}{0.5 + 0.3}(0.5 \cdot 0.8 + 0.3 \cdot 0.5) = 0.69$$

The normalization factor was calculated based on the sum of the weights of all extractors that delivered a value for the field. We call the modified combination method $CombscoreLift$ as empty values cause other values to be lifted during combination. The example shows that $Combscore$ would prefer "TUD" as the best value for the field while $CombscoreLift$ would rank "DocuWare" higher.

To clarify the effect of the two variants, we did an evaluation on our combination of E_2 and E_3 calculating $F1_{Combiner}$ as in Equation (3). The combination with $Combscore$ achieved an F1-measure of 0.821 while the lifting approach with $CombscoreLift$ was slightly worse with $F1_{Combiner}$ of 0.804. But these results are quite domain-specific as they depend on how often extractors return empty results. Thus, for any set of extractors E to be combined using our approach, both methods should be evaluated.

Now the final step of the combination is the optimization of weights. For this purpose, we try out different combinations of weights and measure $F1_{Combiner}$ for each combination. More specifically we can measure F1 for each field and thus decide per field which one is the best combination of weights (see Example (1) for an example distribution of weights). We made evaluation runs with the weights $\{0, 1, 2, 3, 4\}$ resulting in an optimized weight set with only these values. This weight set made up an improvement of 3.3% of $F1_{Combiner}$ compared to equal weights (all weights are set to 1).

Table 4: Summary of results

	E_1	E_2	E_3	E_3 norm.	$Combscore$	$CombscoreLift$
F1 score	0.159	0.589	0.721	0.780	0.821	0.804
Combination gain					0.041	0.024
F1 score in starting phase		0.655	0.576		0.761	

Table 4 summarizes all results for our combination problem. With the $Combscore$ method we reach 4.1% more F1 score than the normalized version of E_3. We thus reached the goal stated in Equation (4). Nevertheless, the biggest effect came out of the normalization of E_3 which boosted F1 by nearly 6%. Alltogether we reached a 10% improvement by the steps discussed in this paper compared to running the original version of E_3 alone.

The last row of Table 4 shows the results of the starting phase, i.e., only the first 100 documents of the evaluation. As we always start with an empty training set, the results of E_3 in the starting phase are expectedly worse than the total F1 score for the whole document set. Here, the combination shows a remarkable improvement of 10% compared to the performance of E_2 and nearly 20% compared to running E_3 alone. So a combination of rule-based and training-based extractors certainly pays off in the starting phase.

8. CONCLUSIONS

This paper dealt with the question, whether a combination of training-based and rule-based extractors for index data extraction from business documents is possible and useful. We have proven the added value of rule-based extractors especially in the starting phase. A good combination of training-based and rule-based extraction seems possible if selection and score normalization are carried out before the actual combination of the single extractors. The approach and measures defined in this paper are a helpful framework for any similar combination attempt. The potential gain of the combination heavily depends on the coverage of the results of the single extractors.

We plan to extend the work on combination of results to the problem of distributed information extraction in the future. Our current research deals with improving the user experience of layout-based extraction by connecting different domain-specific extractors with parent extractors that only deal with the documents not recognized by the child extractors. The question how to combine parent extractor results with child extractor results is still an open challenge in this scenario. Furthermore, it would be interesting to see how our approach is helpful for other domains, e.g., information extraction from the Web or bio-medical use cases.

ACKNOWLEDGEMENTS

This research and project was funded by the German Federal Ministry of Education and Research (BMBF) within the research program "KMU Innovativ" (fund number 01/S12017). We thank our project partners from DocuWare (Michael Berger, Christoph Weidling, Kamil Aliyev) for insightful discussions and for providing us with the document corpus used for the evaluation.

REFERENCES

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

[2] Zhang, L., Pan, Y., and Zhang, T., "Focused named entity recognition using machine learning," in [*Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*], *SIGIR '04*, 281–288 (2004).

[3] Bunescu, R., Ge, R., Kate, R. J., Marcotte, E. M., Mooney, R. J., Ramani, A. K., and Wong, Y. W., "Comparative experiments on learning information extractors for proteins and their interactions," (2004).

[4] Gulan, S. and Fernau, H., "An optimal construction of finite automata from regular expressions," in [*IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2008)*], Hariharan, R., Mukund, M., and Vinay, V., eds., *Leibniz International Proceedings in Informatics (LIPIcs)* **2**, 211–222, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany (2008).

[5] Hu, J., Kashi, R., and Wilfong, G., "Comparison and classification of documents based on layout similarity," *Information Retrieval* **2**(2), 227–243 (2000).

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

[7] Sarkar, P., Society, I. C., Nagy, G., and Fellow, L., "Style consistent classification of isogenous patterns," *IEEE Trans. PAMI* **27**, 88–98 (2005).

[8] Bart, E. and Sarkar, P., "Information extraction by finding repeated structure," in [*Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*], *DAS '10*, 175–182 (2010).

[9] Hanke, M., Muthmann, K., Schuster, D., Schill, A., Aliyev, K., and Berger, M., "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*], (2012).

[10] Kittler, J., Hatef, M., Duin, R., and Matas, J., "On combining classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20**(3), 226 –239 (1998).

[11] Ho, T. K., Hull, J., and Srihari, S., "Decision combination in multiple classifier systems," *IEEE Trans. Pattern Analysis and Machine Intelligence* **16**(1), 66 –75 (1994).

[12] Windridge, D. and Kittler, J., "Combined classifier optimisation via feature selection," in [*Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition*], 687–695, Springer-Verlag, London, UK (2000).

[13] Chen, B. and Zhang, H.-X., "An approach of multiple classifiers ensemble based on feature selection," in [*Proceedings of the 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery - Volume 02*], 390–394 (2008).

[14] Denaux, T., "A k-nearest neighbor classification rule based on dempster-shafer theory," *IEEE Transactions on Systems, Man and Cybernetics* **25**, 804–813 (1995).

[15] Denaux, T., "A neural network classifier based on dempster-shafer theory," (2000).

[16] Giacinto, G. and Roli, F., "Dynamic classifier selection based on multiple classifier behaviour," *Pattern Recognition* **34**, 1879–1881 (2001).

[17] Dempster, A. P., "A generalization of bayesian inference," *Journal of the Royal Statistical Society. Series B (Methodological)* **30**(2), pp. 205–247 (1968).

[18] Srihari, R., Li, W., Cornell, T., and Niu, C., "Infoxtract: A customizable intermediate level information extraction engine," *Nat. Lang. Eng.* **14**(1), 33–69 (2008).

[19] Srihari, R., Niu, C., and Li, W., "A hybrid approach for named entity and sub-type tagging," in [*Proceedings of the sixth conference on Applied natural language processing*], *ANLC '00*, 247–254, Association for Computational Linguistics, Stroudsburg, PA, USA (2000).