Metrics for Complete Evaluation of OCR Performance

Romain Karpinski, Devashish Lohani, Abdel Belaid

Metrics for Complete Evaluation of OCR Performance

R. Karpinski, D. Lohani, and A. Belaïd
Université de Lorraine - LORIA, Vandoeuvre-Lès-Nancy, France
{romain.karpinski,lohani.devashish,abdel.belaid}@loria.fr

Abstract— *In this paper, we study metrics for evaluating OCR performance both in terms of physical segmentation and in terms of textual content recognition. These metrics rely on the OCR output (hypothesis) and the reference (also called ground truth) input format. Two evaluation criteria are considered: the quality of segmentation and the character recognition rate. Three pairs of input formats are selected among two types of inputs: text only (text) and text with spatial information (xml). These pairs of inputs reference-to-hypothesis are: 1) text-to-text, 2) xml-to-xml and 3) text-to-xml. For the text-to-text pair, we selected the RETAS method to perform experiments and show its limits. Regarding text-to-xml, a new method based on unique word anchors is proposed to solve the problem of aligning texts with different information. We define the ZoneMapAltCnt metric for the xml-to-xml approach and show that it offers the most reliable and complete evaluation compared to the other two. Open source OCRs like Tesseract and OCRopus are selected to perform experiments. The datasets used are a collection of documents from the ISTEX*[1] *document database, from French newspaper "Le Nouvel Observateur" as well as invoices and administrative document gathered from different collaborations.*

1. Introduction

Retro-conversion is the task of extracting and recognizing the content of a document image. To perform this task, most OCRs usually work in two steps: region-based text segmentation, and characters recognition using the extracted regions. Depending on the segmentation algorithm used, the location of the lines, words and characters can be obtained. Then, a recognition process can be performed on each character, on a whole word or line depending on the algorithm used. For documents with a simple layout, such as pages in a book, the performance of current OCRs makes it possible to retro-translate them reliably. However, for documents with a much more complex layout, such as newspapers or administrative documents, more segmentation errors may occur.

As shown by Figure 1, segmentation errors can either be a split (under-segmentation) or a merge (over-segmentation) and in two directions: vertical or horizontal. The left side of the figure represents the reference line's structure in the image, while the right side regroups segmentation errors. One can see that some segmentation errors are more severe than others. Segmentation errors are classified in two types:

- Physical: vertical split and merge are severe because they modify the physical structure of the detected content which decreases the character recognition rate.
- Logical: horizontal split and merge do not modify the physical structure, but produce a logical error. They do not perturb the recognition process but may modify the line order.

It is often easier to merge over segmented lines than splitting a line in two. Therefore, splits are less severe than merges.

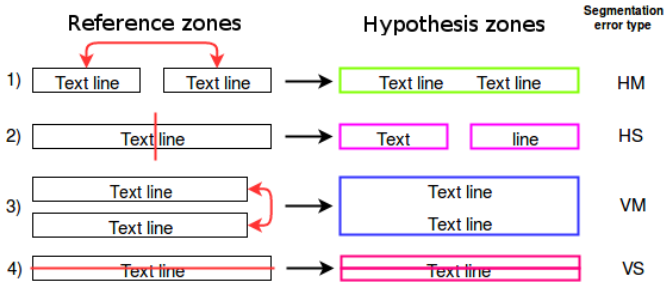

Fig. 1: Segmentation errors and their effects on textual contents. HM is horizontal merge. HS is horizontal split. VM is vertical merge. VS is vertical split.

Regarding the evaluation of character recognition rate, the textual content of reference must be available to align it with the text given by the OCR. The goal of string alignment is to coincide two strings together by making them the same length. Once the strings have been aligned, the character recognition rate can be computed by comparing characters. The evaluation of segmentation quality is often determined by zone alignments. This alignment can be performed on the textual content or using the position. This kind of evaluation is much easier when the spatial information is available. By cons, it is difficult, with only two Texts, to fully evaluate the segmentation.

Thus, alignment methods depend on the format of the reference and the output of the OCR. In our case, no matter what is the input format, the line breaks are accessible. Two input formats are distinguished for the alignment method: text and xml. The text format is a sequence of strings

[1]http://www.istex.fr/, The ISTEX project is a vast program for the acquisition of scientific resources aimed at creating a digital library with the best international standards, accessible remotely by all members of institutions of higher education and research.

representing the sentences of the document image. The xml format is a text with spatial information such as line, word or character bounding boxes. The latter format can be specific to the OCR or a standard format such as hOCR [5] or ALTO [1]. For each pair of inputs (Text-to-Text, Text-to-XML and XML-to-XML), a specific alignment method must be used to produce the most precise evaluation in adequacy with the available information. In the next subsections, the context and issues of each set of inputs will be detailed.

This article is organized as follows. Firstly, we will present in section 2, the related work on the evaluation of character recognition and the quality of segmentation for each set of inputs. Secondly, in section 3, we will explain the selected and our proposed methods. Finally, we will describe in section 4, the datasets and metrics used to evaluate the three methods and conclude in section 5.

2. Related work

2.1 Text-to-Text evaluation

Text-to-Text evaluation consists of aligning and comparing two texts. It is a question of comparing two sets of lines and making the correspondence between them before measuring the error.

The lines may have undergone all the segmentation errors defined above. It can thus be seen that the difficulty of aligning the lines stems directly from the quality of the segmentation. The main difficulty here is that the segmentation needs to be known to align the text and the text needs to be aligned to obtain the segmentation. Therefore, these two tasks must be performed, if possible, jointly.

To carry out text-to-text alignments, Feng et al.[8] proposed to use reference texts from the Gutenberg project. They used unique words to find anchors between the original and the OCR Text. Segments are then obtained between two unique words. Then, an HMM (Hidden Markov Model) aligns the segments at word and character levels. This method requires a correct string sequence, otherwise the segments will not match properly. Yalniz et al.[15] proposed a method named RETAS. The authors try to align texts by spotting unique words common in both texts. Segments can be found between two unique words. Then, they use the longest common substring to align content within segments. Each segment can be viewed as a new document in which one can repeat all the previous steps. Azawi et al.[2] uses a Weighted Finite State Transducer (WFST) framework [3] to align reference text with an incomplete output text of historical documents.

2.2 XML-to-XML evaluation

For xml-to-xml evaluation, zones from both xmls must be aligned together. This alignment will make it possible to compare the zones (error amount and type of errors), and thus the evaluation of the segmentation. Then, this spatial alignment of zones can help with the content alignment (lines, words, etc.). These two steps can be performed separately or jointly. This last step will allow us to evaluate the recognition of the content. Wolf and Jolion [14] proposed a framework for evaluating the position of text lines. The principle of this method is that the evaluation is done at the object level (bounding boxes) and takes into account the quality of each correspondence between the hypothesis boxes and the reference ones. Shafait et al. [13] proposed a pixel-based algorithm. They calculate a weighted bipartite graph called pixel-correspondence graph where each node represents a segmentation. To accomplish zone to zone alignments, Seo et al. [12] offers several pixel based metrics. In the ICDAR competition, Clausner et al. [6] and Antonacopoulos et al. [4] represent zones as polygons and polygons as a set of intervals. Computations are faster than pixel based approach. Galibert et al. [10] propose to use the ZoneMap metric. They compute a link strength between intersecting bounding box. Then groups are formed iteratively using the links. Many-to-many relations in a group cannot be made. The cardinality of each group determines the segmentation error. A score named $E_{ZoneMap}$ is computed using the segmentation error type of concerned areas combined with the classification error.

2.3 Text-to-XML evaluation

The problem raised by this case is a combination of the two cases previously presented. It is very similar to text-to-text alignment problem with in addition, a spatial information on the output of the OCR. The format of the reference is a text and the OCR provides us an xml file indicating the location of the text zones. The reference content must be aligned to both segmentation and output content from the OCR. As for text-to-text alignment, it is difficult to align the content without knowing the quality of the segmentation and reciprocally, it is difficult to evaluate the quality of the segmentation without aligning the content. The only reliable information on the segmentation is given by the line breaks in the reference text.

Kornfield et al.[11] proposed to use dynamic time warping (DTW) to align handwritten document word by word to their transcripts. They segment the image to obtain bounding boxes. Then, ASCII words and bounding boxes form two time series whose distance must be minimized. For each series, elements are sorted from their order of appearance. The maximum allowed warping is controlled by a path constraint. The overall distance is computed using the positions order combined with a custom distance between the bounding box and word.

Fischer et al.[9] employ an HMM-based alignment to align historical documents with a reference string. Text recognition of line images is performed to estimate the word's location in a single line. Then, they carry out the alignment of the transcript and recognized text lines using

the edit distance. Finally, they spot keywords to find and fill gaps due to inaccurate text recognition.

3. Our contribution

Our contribution is divided into 4 parts:

- Three methods using three different types of inputs (Text-to-Text, Text-to-XML and XML-to-XML) are defined and their performances compared on the same dataset.
- We studied the impact on performance for the three methods using two categories of documents (1D and 2D).
- A new approach to solve the Text-to-XML alignment problem is proposed to take into account different line orders as well as serious segmentation errors.
- A detailled analysis of the XML-to-XML method ZoneMapAltCnt is performed and we show that it is able to spot the strengths and weaknesses of the tested algorithm (the OCR).

3.1 ZoneMapAltCnt

ZoneMapAltCnt is a new method that we propose here for complete ocr evaluation. It is an extension of ZoneMapAlt [16] where the content of zones in different configurations is taken into account. It provides an unified evaluation criteria for the OCR evaluation. It takes as input, the various configuration groups with their RZs, HZs and contents produced by the ZoneMapAlt, considering word as a zone and outputs word and character accuracy metrics as shown in the following sections.

3.1.1 Character metric

Character metric is based on the Levenshtein distance which represent the minimum number of edit operations (character insertions, deletions, and substitutions) needed to correct the hypothesis Text (OCR generated Text) to match it with the reference Text. Let C_{ins} be the number of character insertions in HZ to match with RZ, C_{del} be the number of char deletions and C_{sub} be the number of char substitutions. Let C_H be the number of characters in HZ and C_R be the number of characters in RZ. The total character error C_{error} is defined by the Equation 1:

$$C_{error} = C_{ins} + C_{del} + C_{sub} \tag{1}$$

The number of correct characters $C_{correct}$ is given as:

$$C_{correct} = C_{aln} - C_{error} \tag{2}$$

where C_{aln} is the number of characters in the aligned sequence.

Character precision and recall are computed respectively as defined in the equation 3a and equation 3b.

$$C_{Precision} = \frac{C_{correct}}{C_H} \tag{3a}$$

$$C_{Recall} = \frac{C_{correct}}{C_R} \tag{3b}$$

The matching of hypothesis Text with reference Text is performed on word level zones and depends on the configuration of RZ and HZ. It will be described in the subsection 3.1.4.

3.1.2 Word metric

Word metric is based on the Levenshtein distance, like in characters, but at a word level. This means that we will look for the minimum number of word insertions, deletions, and substitutions needed to correct the hypothesis Text to match it with the reference Text. If word in RZ is a substring of the word in corresponding HZ, then it is considered correct. The number of erroneous words W_{error}, the number of correct words $W_{correct}$, the word precision $W_{Precision}$ and the word recall W_{Recall} are computed similarly like in the character metric.

3.1.3 Strict word metric

The strict word metric is identical to word metric except for the computation of a correct word. Strictly correct word $W^S_{correct}$ is a word in RZ which matches all its characters in corresponding HZ such that there are no additional characters in HZ, i.e., the word is RZ should exactly match the word in HZ and should not just be a substring of HZ.

3.1.4 Metrics computation by group configuration

This section will provide details on the computation of character errors (C_{del}, C_{sub}, C_{ins}), word errors (W_{del}, W_{sub}, W_{ins}) and strictly correct words ($W^S_{correct}$) depending on the group configuration.

1) False Alarm: The group contains only one HZ with content and no corresponding RZ. It means that the system has over-detected this zone and its content.

$$C_{del} = C_{del} + \#chars(HZ) \tag{4a}$$

$$W_{del} = W_{del} + 1 \tag{4b}$$

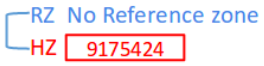

Fig. 2: Class: False alarm.

2) Miss: The group contains only one RZ with content and no corresponding HZ. It means that the system has under-detected this zone and its content.

$$C_{ins} = C_{ins} + \#chars(RZ) \tag{5a}$$

$$W_{ins} = W_{ins} + 1 \tag{5b}$$

3) Match: The group contains only one HZ and one RZ. Compare content of RZ with HZ.

$$C_{ins}, C_{del}, C_{sub} = LD(HZ, RZ) \tag{6a}$$

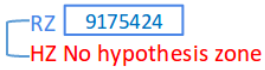

Fig. 3: Class: Miss.

where $LD(x, y)$ is the Levenshtein distance from source string 'x' to the destination string 'y'.
If word in RZ is exactly equal to word in HZ,

$$W^S_{correct} = W^S_{correct} + 1 \tag{6b}$$

If word in RZ is not a substring of word in HZ, then

$$W_{sub} = W_{sub} + 1 \tag{6c}$$

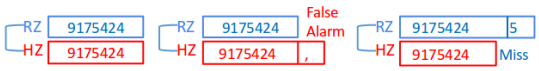

Fig. 4: Class: Match.

4) Split: The group contains only one RZ and more than one HZs. It means that the RZ has been segmented into several parts.

$$C_{ins}, C_{del}, C_{sub} = LD(H_{con}, RZ) \tag{7a}$$

where H_{con} is the concatenated content from HZs.
If word in RZ is equal to a word in any of HZs, then

$$W^S_{correct} = W^S_{correct} + 1 \tag{7b}$$

If word in RZ is not a substring of H_{con}, then

$$W_{sub} = W_{sub} + 1 \tag{7c}$$

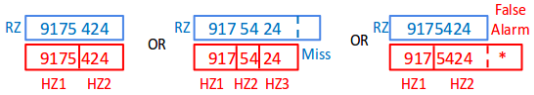

Fig. 5: Class: Split.

5) Merge: The group contains more than one RZs and one HZ, meaning RZs have been merged by an HZ.

$$C_{ins}, C_{del}, C_{sub} = LD(HZ, R_{con}) \tag{8a}$$

where R_{con} is the concatenated content from RZs.
For each RZ_i in RZs:
If word in RZ_i is equal to a word in HZ, then

$$W^S_{correct} = W^S_{correct} + 1 \tag{8b}$$

If word in RZ_i is not a substring of word in HZ, then

$$W_{sub} = W_{sub} + 1 \tag{8c}$$

6) Multiple: The group contains more than one RZs and HZs.

$$C_{ins}, C_{del}, C_{sub} = LD(H_{con}, R_{con}) \tag{9a}$$

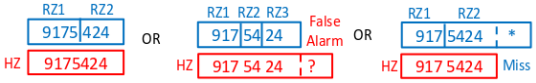

Fig. 6: Class: Merge.

For each RZ_i in RZs:
If word in RZ_i is equal to a word in any of HZs, then

$$W^S_{correct} = W^S_{correct} + 1 \tag{9b}$$

If word in RZ_i is not a substring of H_{con}, then

$$W_{sub} = W_{sub} + 1 \tag{9c}$$

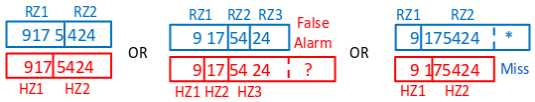

Fig. 7: Class: Multiple.

3.2 Text-to-xml Approach

We propose a new approach to solve the problem of aligning a text with an inaccurate segmentation and its content. This method is a mix between the text and zone alignment methods. For text-to-xml alignments, the main issues that appears are the modification of the line order and of the physical structure. The method needs to take these errors into account. Let the set of reference lines: $R = \{r_0, r_1, ..., r_i\}, i \in \mathbb{N}$. Let the set of hypothesis lines: $H = \{h_0, h_1, ..., h_j\}, j \in \mathbb{N}$. The Algorithm 1 describes the main steps of the proposed method.

Algorithm 1: Text-to-xml - Alignment Algorithm

```
begin
    MarkUniqueStrings(R, H)
    DetectAndCorrectMergesSplits(R, H)
    Boolean matched ⟵ true
    while matched do
        CorrectLastAlignments(R, H)
        matched ⟵ AlignNewElements(R, H)
    end
end
```

3.2.1 Marking of common unique strings

We use a parameter n representing a number of consecutive words tested for the unique strings. We decrease n until one because having n consecutive unique words is more reliable than one unique word. A limitation of this marking is that an hypothesis word can be modified to match another word in reference text resulting in a false alignment. The probability to obtain a false match between

two unique words is decreasing when the size of the word increases. The goal of this step is to find the most lines having a single string. Following the marking of words, a so-called "natural" alignment emerges. When two strings are associated together, two lines are also associated. This first treatment provides us with an alignment draft that must be checked and which will serve as a basis for realizing the entire alignment.

3.2.2 Detection and correction of merges and splits

After the previous step, an alignment draft is obtained. This draft may have segmentation errors such as merges and splits. If an hypothesis line is aligned with at least 2 reference lines then this is a merge. Conversely, if one reference line is aligned with at least 2 hypothesis lines then it is a split.

- In the case of a merge, the hypothesis line must be cut and realigned with the reference lines with which it is aligned. To restore the merge, we iterate on each of the reference lines matched to the hypothesis line. Two lines are taken and an attempt to match their contents is being made by rejecting one end (align to left or right). If the extremity is not empty, then the two preceding strings are matched and the extremity becomes a separate line. If the end is empty, then all content has been aligned. It should be noted that when cutting an element of the hypothesis, an estimation of the cut location is performed by estimating the space of a character in the line.
- In the case of a split, the hypothesis lines matched to the concerned reference line must be merged together. The split is a horizontal split because in the case of a vertical split, the text would not be correct since the later modifies the content. Since the split is horizontal, we must order the lines from left to right and merge them. Before merging, we must verify that the associated lines are not due to a false matching between unique words. To do so, we check if the total size of the assembled hypothesis lines is close to the size of the reference line. We fix a threshold of 20% difference tolerance between the assembled string and the reference one. If the threshold is respected, then we merge iteratively the hypothesis lines from left to right.

3.2.3 Correction of last alignments

This step checks that one line has not been subject to a merge or split in the same way than previously. The only difference is that here, we do not know which lines are concerned by these segmentation errors. A cardinality relation of one-to-one does not mean that there are no segmentation errors. It is important to check each alignment that we have made. We use the same process as the previous step with the exception that when a merge is detected, we cut parts that match and create a new line from the remaining. The technique is illustrated in the pseudo-code Algorithm 2.

Algorithm 2: Text-to-xml - Correction of last alignments

```
1  begin
2    for r_i ∈ R do
3      if !r_i.checked then
4        Ligne remaining for h_k ∈ r_i.matches
           do
5          remaining ⟵ matchLines(r_i, h_k)
6          if !remaining.empty then
7            h_k.content ⟵ remaining
8          end
9        end
10       g_i.checked ⟵ true if
           !remaining.empty then
11         addUnmatchedLine(H, remaining)
12       end
13     end
14   end
15 end
```

3.2.4 Alignment of new elements

We iterate on the reference lines by making intervals between two lines. These intervals form lists of non-aligned reference lines. Also, it gives us, thanks to the matches of the two lines forming the interval, a search zone. This zone corresponds to the rectangle formed between the two matches. The area thus defined allows us to use the positions of the hypothesis lines in order to select them to align them. We have two sets of lines that we must compare to associate them. This last step is performed by a recursive method of searching for the best correspondence between the lines. This allows us to find an alignment respecting the order of origin of the lines while rejecting if necessary certain lines whose content is too far. The previous step called correction of last alignments and this step are performed while new elements from this step are created.

4. Results and Experiments

4.1 Datasets

Two corpora are used to perform the experiments. The first corpus D_1 is composed of 49 documents of 1D type and is coming from book or mono-page articles. These are documents with a very simple layout with only little variations. They come from several editors and from several sources. There are scientific articles in French, English, German and in Spanish from 1942 to 2000. Some documents contain graphical zones such as tables and figures.

The second corpus D_2 is composed of 30 multicolumn documents with headers, footnotes and references. The layout of theses document can be a mix of 1D sections and

2D sections. All documents have been manually annotated at line level with their content.

4.2 Performance comparison between 1D and 2D documents

The goal of this experiment is to put the method RETAS to the test by comparing the results from a dataset of 1D documents against 2D documents. The theoretical performance has been obtained by manually aligning the contents before evaluating the character error rate. The OCR OCRopus has been used to conduct the experiment. It appears that this OCR often makes segmentation errors with 2D documents such as merging horizontal lines belonging to two separate Text columns. Thus, it makes this OCR a good candidate to evaluate the performances of the RETAS method on 1D and 2D documents. Results of this experiment are presented in Table 1. The method works well for 1D documents with a performance gap of 0.64% which can be considered as negligible in this context. However, the RETAS method is clearly not working properly with 2D documents with a performance gap of almost 25%. In this context, the difference in performance cannot be considered to be correct. If we consider this method to represent the Text-to-Text alignment schemes, then they are not a good fit to evaluate reliably 2D documents. The reason to this is that these method do not take into account segmentation errors that can happen in 2D documents.

Corpus	Theoretical Performance	Real Performance	Variation
1	96.44%	95.80%	-0.64%
2	96.40%	71.75%	**-24.65%**

Table 1: Performance results comparison of the RETAS method on the D_1 and D_2 corpora.

4.3 Comparing the three types of inputs

We have seen that RETAS is not suited to evaluate properly 2D documents. This mean that generally, Text-to-Text methods will fail to evaluate 2D documents properly because they may contain segmentation errors that were not present in 1D documents. The next experiment aims to determine which method is best between Text-to-XML and XML-to-XML. Only the content evaluation can be compared with the three methods. Even if Text-to-XML is theoretically able to spot some segmentation errors, it is not able to do it reliably because of the lack of information. This is the reason why we only compare the character recognition performances to determine which method is the best one. A comparison of performances is shown in Table 2. A clear improvement can be observed between the Text-to-Text method and the other two. The most reliable method is ZoneMapAltCnt since it is able to be really close (0.12% gap) to the theoretical performance. Regarding the Text-to-XML approach, it is not very far from the ZoneMapAltCnt performances, however it is incapable of providing a detailled analisis of segmentation errors. The main errors occur when the physical structure of the document is complex and we do not find the right hypothesis candidates to align with the reference. It should also be noted that ZoneMapAltCnt is supposed to work initially on word but also works on lines. The alignment is therefore less precise at line level than it is at word level but precise enough to be considered as correct. The performance gap can be explain by ambiguous cases where the alignment is difficult to perform. In the case of a vertical merge where the content has not been properly recognized, it is difficult to compare in only one way two reference lines with one hypothesis line for which the content is wrong. The string comparison will not be equal to the one of the reference but can be considered correct too.

Method	Theoretical Performance	Real Performance	Variation
RETAS	96.40%	71.75%	-24.65%
Text-to-XML	96.40%	96.03%	**-0.37%**
ZoneMapAltCnt	96.40%	96.28%	**-0.12%**

Table 2: Performance results of the three methods on the 2D documents using the OCRopus OCR.

4.4 A detailed evaluation with ZoneMapAltCnt

To show the ability of ZoneMapAltCnt method to understand the behavior of an OCR we need to have the ground truth at word level. Since D_1 and D_2 are annotated at line level, we choose to use another dataset already annotated at word level. This corpus D_3 is composed of 30 administrative documents such as invoices or contracts. Results of Tesseract OCR 3.05 and 4.0 on the D_3 corpora is presented in Table 3. In the segmentation section, the match value represent how well the total reference area was matched. From Miss to Multiple, each segmentation error type denotes its error contribution to the total segmentation error. The ZoneMap$_{score}$ is the total error divided by the total reference area. Looking only at matches, the performance is greater for 4.0 than for 3.05. This means that the segmentation of 4.0 is more precise than for 3.05. It can be observed that for every corpus, false alarm error is responsible for at least 96% of the error score. Even after this extremely dominating segmentation error, character metric results are surprisingly excellent with at least 93.53% precision across all corpora. This led us to formulate the following hypothesis: most of the false alarm errors does not contain any character otherwise the character precision would have been greatly affected. To verify this hypothesis, a visual inspection has been conducted which reveals that elements constituting graphic zones such as table lines are identified as zones without content. Hence we confirmed that this hypothesis is true and the method lead us to discover an important behavior of Tesseract OCR. Another important aspect of OCR that needs investigation is comparison between word and strict word metrics. Comparison of W_{Recall} and W^S_{Recall} reveals the percentage of

OCR	Segmentation							Recognition					
	Match	Miss	False Alarm	Split	Merge	Multiple	ZoneMap$_{score}$	C_{Recall}	$C_{Precision}$	W_{Recall}	$W_{Precision}$	W^S_{Recall}	$W^S_{Precision}$
Tesseract 4.0	97.52% (#5872)	0.43% (#129)	98.65% (#988)	0.40% (#47)	0.43% (#148)	0,07% (#42)	251.92	98.10%	98.23%	95.47%	94.64%	90.43%	89.64%
Tesseract 3.05	95.79% (#5808)	0.55% (#158)	96.83% (#1250)	0.69% (#165)	1.86% (#216)	0.06% (#15)	210.54	94.33%	93.53%	87.49%	83.46%	82.4%	78.6%

Table 3: Main results of ZoneMapAltCnt algorithm with Tesseract OCR version 4.0 and 3.05 on the D_3 corpus.

words that could have been retrieved if the segmentation was correct or corrected. W_{Recall} shows how well the words are recognized with a tolerance to segmentation errors while W^S_{Recall} expresses how well the words are recognized as they are segmented. Difference between word and strict word metrics explains how the errors are distributed among words and how the segmentation errors affect the word recognition. The main observation that can be made is that 4.0 outperforms 3.05 in each metric. However, the percentage of words($W_{Recall} - W^S_{Recall}$) that could have been possibly retrieved is approximately the same for each version (5.04% for 4.0 and 5,09%). This implies that even if the segmentation algorithm has been improved (increase of matched zones and reduction of segmentation errors) there is still 5.04% of performance to gain by improving further the segmentation. Thanks to ZoneMapAltCnt metric, we are able to explain the relationship between zone segmentation and content recognition of the OCR which otherwise was not possible to discover.

5. Conclusion

In this paper, we proposed to study three types of inputs and their corresponding methods for performing segmentation and character recognition evaluations on OCRs. The method representing text-to-text appoach, RETAS, is not suited to handle 2D documents as its results are distant from the reference (24.65%). A new method proposed to meet the particular constraints of the text-to-xml approach, which is a compromise between text and text with its spatial representation. A huge performance improvment of more than 24% has been observed with this method. We found that the method is working well when the segmentation and the content allow placing reliable marks. This method can be effectively used to align content by considering possible segmentation errors when disposing only of a reference text. To represent the xml-to-xml approach, the ZoneMapAltCnt method is introduced. This method uses the reliable result of the ZoneMapAlt algorithm to match the contents together and thus takes into account the segmentation errors. It also allows us to perform a joint evaluation of document zone segmentation and textual content. The xml-to-xml approach is giving the most detailed evaluation of the 3 input types and is able to evaluate the segmentation as well as the recognition performances. In future work, we plan on unifying the methods by working on a single method that will behave accordingly to the information available.

References

[1] Alto XML standard. http://www.loc.gov/standards/alto/.

[2] M. Al Azawi, M. Liwicki, and T. M. Breuel. Wfst-based ground truth alignment for difficult historical documents with text modification and layout variations. In *IS&T/SPIE Electronic Imaging*, pages 865818–865818. International Society for Optics and Photonics, 2013.

[3] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri. Openfst: A general and efficient weighted finite-state transducer library. In *International Conference on Implementation and Application of Automata*, pages 11–23. Springer, 2007.

[4] A. Antonacopoulos and D. Bridson. Performance analysis framework for layout analysis methods. In *Document Analysis and Recognition (ICDAR).*, volume 2, pages 1258–1262. IEEE, 2007.

[5] T. M. Breuel. The hocr microformat for ocr workflow and results. In *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on*, volume 2, pages 1063–1067. IEEE, 2007.

[6] C. Clausner, S. Pletschacher, and A. Antonacopoulos. Scenario driven in-depth performance evaluation of document layout analysis methods. In *Document Analysis and Recognition (ICDAR).*, pages 1404–1408. IEEE, 2011.

[7] A. Farahmand, H. Sarrafzadeh, and J. Shanbehzadeh. Document image noises and removal methods. 2013.

[8] S. Feng and R. Manmatha. A hierarchical, hmm-based automatic evaluation of ocr accuracy for a digital library of books. In *Digital Libraries, 2006. JCDL'06. Proceedings of the 6th ACM/IEEE-CS Joint Conference on*, pages 109–118. IEEE, 2006.

[9] A. Fischer, E. Indermühle, V. Frinken, and H. Bunke. Hmm-based alignment of inaccurate transcriptions for historical documents. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 53–57. IEEE, 2011.

[10] O. Galibert, J. Kahn, and I. Oparin. The zonemap metric for page segmentation and area classification in scanned documents. In *Image Processing (ICIP).*, pages 2594–2598. IEEE, 2014.

[11] E. M. Kornfield, R. Manmatha, and J. Allan. Text alignment with handwritten documents. In *Document Image Analysis for Libraries, 2004. Proceedings. First International Workshop on*, pages 195–209. IEEE, 2004.

[12] W. Seo, M. Agrawal, and D. Doermann. Performance evaluation tools for zone segmentation and classification (pets). In *Pattern Recognition (ICPR).*, pages 503–506. IEEE, 2010.

[13] F. Shafait, D. Keysers, and T. M. Breuel. Pixel-accurate representation and evaluation of page segmentation in document images. In *Pattern Recognition (ICPR).*, volume 1, pages 872–875. IEEE, 2006.

[14] C. Wolf and J.-M. Jolion. Object count/area graphs for the evaluation of object detection and segmentation algorithms. *International Journal of Document Analysis and Recognition (IJDAR).*, 8(4):280–296, 2006.

[15] I. Z. Yalniz and R. Manmatha. A fast alignment scheme for automatic ocr evaluation of books. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 754–758. IEEE, 2011.

[16] R. Karpinski and A. Belaid. ZoneMapAlt: An alternative to the ZoneMap metric for zone segmentation and classification. In *DAS, Vienna, Austria*, 2018.