End-To-End Measure for Text Recognition

Gundram Leifert, Roger Labahn
Computational Intelligence Technology Lab
University of Rostock
18057 Rostock, Germany
{gundram.leifert,roger.labahn}@uni-rostock.de

Tobias Grüning, Svenja Leifert
PLANET artificial intelligence GmbH
Warnowufer 60
18057 Rostock, Germany
{tobias.gruening,svenja.leifert}@planet.de

Abstract—Measuring the performance of text recognition and text line detection engines is an important step to objectively compare systems and their configuration. There exist well-established measures for both tasks separately. However, there is no sophisticated evaluation scheme to measure the quality of a combined text line detection and text recognition system. The F-measure on word level is a well-known methodology, which is sometimes used in this context. Nevertheless, it does not take into account the alignment of hypothesis and ground truth text and can lead to deceptive results. Since users of automatic information retrieval pipelines in the context of text recognition are mainly interested in the end-to-end performance of a given system, there is a strong need for such a measure. Hence, we present a measure to evaluate the quality of an end-to-end text recognition system. The basis for this measure is the well established and widely used character error rate, which is limited – in its original form – to aligned hypothesis and ground truth texts. The proposed measure is flexible in a way that it can be configured to penalize different reading orders between the hypothesis and ground truth and can take into account the geometric position of the text lines. Additionally, it can ignore over- and under-segmentation of text lines. With these parameters it is possible to get a measure fitting best to its own needs.

Index Terms—measure, end-to-end, character error rate, word error rate, F-measure, bag-of-word, HTR

I. INTRODUCTION

Finding and reading textual information in an image is a common task in many real-world scenarios. One application is the transcription of historical documents. Typically, the focus is to transcribe the written text in the semantically correct order, whereas the geometric position of text lines is not in the scope of interest. Another use case is to make a collection searchable, i.e., to allow for keyword spotting. In such a scenario, a system is used to create some kind of index for the whole collection. So the main focus is to find textual information in the image, whereas reading order of the text lines and sometimes even the text position is not of importance. In contrast, there are other applications for which the geometric information of text lines is necessary, e.g., the postal inbox processing for insurances and banks. Their purpose is to automatically read and classify all incoming letters. Often, the input image should be enriched with a layer of textual information. Therefore, geometric positions and the reading order of text lines are important to place the transcribed text at the right position. Having these use cases with entirely different key aspects, there is the demand for a configurable end-to-end evaluation which is adaptable to the specific needs.

In the context of information retrieval the bag-of-word (BOW) measure is widely used [1]. It can be efficiently calculated by splitting the text into words and measuring precision, recall and F-measure of the text. The BOW suffers from three major drawbacks. First, there is no unique definition of how a "word" should look like. This results in inconsistent and incomparable values of the BOW measure for different tokenizations of text lines into words. Second, a wrong character produces an error for the entire word. Comparably, segmentation errors are also penalized quite strongly. An erroneously recognized space character results in two word errors. Third, the BOW is not aware of any (potentially important) reading order and consequently does not penalize any permutation of recognized words.

For the decoupled problems of layout analysis (LA) and handwritten text recognition (HTR) there are well established measures. For the LA, which extracts text lines on pixel level, there are evaluation schemes based on different entities. For instance, based on pixel information [2], baselines [3] or origin points [4]. Each of these schemes has its application area and consequently its right to exist. On the other hand, the standard to evaluate the quality of an HTR system is the character error rate (CER), which has been used for decades. A major drawback of the CER is that it requires two aligned sequences of characters which usually are the transcriptions of text lines. This paper will provide task-dependent solutions for this alignment and an implementation is freely available supporting the well established PageXML format [5].

The paper is structured as follows: Sec. II will derive the end-to-end CER from the classical CER and will motivate and define different configurations of this measure. We will briefly demonstrate how to get from CER to word error rate (WER) and finally to BOW. In Sec. III the calculation of the introduced measures is described and the exactness of the proposed algorithms is proven for certain conditions. A short summary and outlook concludes the paper in Sec. IV.

II. MEASURE FORMULATION

The CER is based on the Levenshtein distance (LD), which counts the character manipulations (insertion, deletion, substitution) to map one string to another [6]. Let $\Sigma$ be the alphabet of all characters and $\Sigma^*$ the Kleene star of $\Sigma$. Let $g_i \in \Sigma$
be the i-th character of $g \in \Sigma^*$ and $g_{i:j} := (g_i, g_{i+1}, \ldots, g_j)$ a subsequence of $g$. In the following it is required that the hypothesis (HYP) and ground truth (GT) $h, g \in \Sigma^*$ do not have leading or trailing spaces. The LD between $h$ and $g$ is defined by recursion. Let

$$
\delta_{i,j} = \begin{cases} 
0 & \text{if } h_i = g_j \\
1 & \text{else}
\end{cases} 
$$

be the function that indicates the difference between $h_i$ and $g_j$. Let $\Delta_{i,j} = LD(h_{1:i}, g_{1:j})$ be the number of manipulations which have to be done on $h_{1:i}$ to map it to $g_{1:j}$. This function is defined recursively over $i$ and $j$ with $[n] := \{1, \ldots, n\}$ as follows

$$
\Delta_{0,0} = 0, \quad \Delta_{i,0} = i \quad \forall i \in [|h|], \quad \Delta_{0,j} = j \quad \forall j \in [|g|] \\
\Delta_{i,j} = \min \left\{ \Delta_{i-1,j-1} + \delta_{i,j}, \Delta_{i-1,j} + 1, \Delta_{i,j-1} + 1 \right\} \forall i \in [|h|], j \in [|g|],
$$

so that we obtain the LD of the strings $h$ and $g$ by

$$
LD(h, g) := LD(h_{1:|h|}, g_{1:|g|}) = \Delta_{|h|,|g|}.
$$

Since $\Delta_{i,j}$ in (2) is recursively defined using values with one step back in $i$ or $j$, this problem can be efficiently solved using dynamic programming over the two-dimensional $i$-$j$-space. Finally, the character error rate $CER : \Sigma^* \times \Sigma^* \rightarrow \mathbb{R}^+$ is defined by

$$
CER(h, g) := \frac{LD(h, g)}{|g|}.
$$

Of note, the CER could exceed 1 and it is not commutative, i.e., $CER(h, g) \neq CER(g, h)$ for certain inputs $g, h$.

To evaluate a system’s performance, the CER is calculated for a certain amount of text lines – the so-called test set – to get a reliable statistic. The test set is a $K$-tuple of GT sequences $G := (\hat{G}_1, \ldots, \hat{G}_K)$, $\hat{G}_k \in \Sigma^*$. The HYP $H := (H_1, \ldots, H_K)$ is calculated by the system which has to be evaluated. The CER for a given test set is defined by

$$
LD(H, G) := \sum_{k=1}^K LD(H_k, \hat{G}_k) \\
|G| := \sum_{k=1}^K |\hat{G}_k| \\
CER(H, G) := \frac{LD(H, G)}{|G|}.
$$

To measure an end-to-end system, the CER calculation has to be done straight-forward by

$$
|G| = \sum_{k=1}^K |\hat{G}_k| = \sum_{k=1}^K \sum_{x=1}^{|\hat{G}_k|} \left| (\hat{G}_k)_x \right|,
$$

whereas the expansion for the numerator

$$
LD(H_k, \hat{G}_k) = \sum_{k=1}^K LD(H_k, \hat{G}_k)
$$

is non-trivial, because it is not clear how to calculate $LD(H_k, \hat{G}_k)$ easily. Different ways to calculate $LD(H, G) := LD(H_k, \hat{G}_k)$ will be proposed and discussed in the following. $H, G \in (\Sigma^*)^K$ are tuples of character sequences, but $|H| \neq |G|$ has to be considered, which means that the numbers of text lines differ (mainly resulting from an erroneously working LA). The key idea is to expand (1) and (2) to match two tuples of character sequences. Let $H := (H_1, \ldots, H_N)$ be the HYP lines and $G := (G_1, \ldots, G_M)$ the GT lines. For the reason of simplicity, we write $H_y \in H$ if a text line belongs to the tuple of text lines, and $H' \subset H \Leftrightarrow \forall H'_y \in H' : H'_y \in H$. The Assignment Matrix $A \in \mathcal{A}$ defines which HYP and GT lines are assigned to each other. We define the set of valid assignment matrices as

$$
\mathcal{A} := \left\{ A \in \{0, 1\}^{N \times M} \mid |A|_1 \leq 1 \land \|A\|_\infty \leq 1 \right\},
$$

whereas $A_{y,x} = 1$ means that $H_y$ and $G_x$ are assigned to each other. The conditions in (3) ensure that each GT line is assigned to at most one HYP line and vice versa. With $A \in \mathcal{A}$ it is possible to define the three sets

$$
W := W(A) = \{(y, x) \in [N] \times [M] \mid A_{y,x} = 1\}, \\
U := U(A) = \{y \in [N] \mid \forall x \in [M] : A_{y,x} = 0\}, \\
V := V(A) = \{x \in [M] \mid \forall y \in [N] : A_{y,x} = 0\},
$$

with $W$ containing the indices of the assigned text lines of $H$ and $G$ whereas $U$ and $V$ contain the indices of the unmatched text lines. Note that all indices are in one of these sets, consequently $|W| + |U| + |V| = N + M$ holds. The minimal LD is then defined by

$$
LD(H, G) := LD_A(H, G) = \min_{A \in \mathcal{A}} \sum_{(y,x) \in W(A)} LD(H_y, G_x) + \sum_{y \in U(A)} |H_y| + \sum_{x \in V(A)} |G_x|,
$$

and the CER is defined by

$$
CER(H, G) = \frac{LD(H, G)}{|G|}.
$$

Of note, the LD in the sum of (4) is the basic LD which operates on single text lines. If $CER(H, G) = 0$ holds, it is obvious that $|G| = N = M = |H|$, $A$ is a permutation matrix and $\forall A_{y,x} = 1 : H_y = G_x$. This also results in empty sets $U$ and $V$.

Next, we will describe different ways to modify this error rate. Whereas Sections II-A and II-B add restrictions for
the LD calculation, Section II-C allows a modification of \( H \) to better match \( G \). In Section II-D we will discuss the combination of these modifications. Finally, a comparison between CER, WER and BOW is given in Section

### A. Penalizing Reading Order Errors

Even if the reading order of pages with tables, notes, marginalia or multiple columns is hard to define, it is crucial for semantic understanding. So it is reasonable to extend the restriction of (3) to

\[
A^R := \{ A \in A \mid \forall y,y' \in [N], \forall x,x' \in [M] : y < y' \land A_{y,x} = A_{y',x'} = 1 \Rightarrow x < x' \}. 
\]

This additional restriction prevents assignments which are not aware of the orders of \( H \) and \( G \), e.g., an assignment for which the first line of \( H \) is assigned to the last one of \( G \) and vice versa.

We focus on the top right four text lines of Fig. 1 to demonstrate the effect for a simple example, i.e.,

\[
H = (H_4, H_7, H_5, H_8) , \quad G = (G_7, G_8, G_{10}, G_{11}).
\]

In this order the HYP and GT only differ in the sorting along columns and rows as well as in one error in the hypothesis \( H_7 \). Without the reading order constraint we get \( LD(A(H, G)) = LD(H_7, G_{10}) = 1 \), because \( W = \{(1,1), (3,2), (2,3), (4,4)\} \) is feasible and there are no errors in three out of the four assigned text lines. In contrast, with the constraint \( A \in A^R \) we get \( W = \{(1,1), (3,2), (4,4)\} \). The assignment \((2,3)\) is not allowed due to the reading order constraint. Consequently, \( H_7 \) and \( G_{10} \) are not assigned, which results in \( U = \{2\}, V = \{3\} \) and \( LD(A(H, G)) = |H_7| + |G_{10}| = 5 \).

Based on (3) we define

\[
LD^R(H, G) := LD(A^R(H, G)) \tag{5}
\]
as minimal LD between \( H \) and \( G \) that penalizes reading order errors and \( CER^R(H, G) := \frac{LD^R(H,G)}{|G|} \).

### B. Using Geometric Information as Restriction

Especially for tables with short text lines containing for instance the age, the birth date or running numbers, it is possible that the minimization of (3) assigns a wrongly transcribed HYP text line to a GT text line which is located at an entirely different position in the image. E.g., \( H_9 = G_{10} \) holds for the text lines of Fig. 1 but their geometric positions do not match. An assignment of such kind could erroneously reduce the CER. Consequently, it makes sense to only allow assignments between \( H_y \) and \( G_x \) if their geometric positions match. Again, the idea is to add restrictions for \( A \), such that two text lines can only be assigned, if they are “(geometrically) close” to each other. There are many possibilities to determine if two text lines are close to each other or not. Here, the well-established method of [3] is used. We say two text lines are close, if their baselines are geometrically close to each other (see Section [II-B for details). Let \( \mathcal{N}(G_x) \subset H \) be the set of all text lines in \( H \), that are close to \( G_x \). We extend (3) to

\[
A^G := \{ A \in A \mid A_{y,x} = 1 \Rightarrow H_y \in \mathcal{N}(G_x) \} \tag{6}
\]

and modify (4) with \( A^G \)

\[
LD^G(H,G) := LD(A^G(H,G)) \tag{7}
\]
to define \( CER^G(H,G) := \frac{LD^G(H,G)}{|G|} \).

### C. Non-Penalizing of Segmentation Errors

If an LA does not detect a text line \( G_x \), the LD increases by \(|G_x|\), as well as the LD increases by \(|H_y|\) for an erroneously detected text line \( H_y \). Even more crucial are falsely merged text lines. For example, the hypothesis \( (H_2) \) of Figure 1 is an erroneously merged text line. For

\[
H = (H_2) , \quad G = (G_2, G_5), \tag{8}
\]

the calculation \( LD(H, G) \) leads to \( U = \emptyset, V = \{2\} \), \( W = \{(1,1)\} \) and \( LD(H, G) = LD(H_2, G_2) + |G_5| = 5+4 = 9 \). The resulting LD could be considered to be quite high based on the fact, that the recognized text is entirely correct, but merged. The same argument is valid for an erroneous split of a text line. Hence, it is meaningful to modify the LD calculation such that it does not penalizes this kind of split and merge errors.

It is assumed that these kind of segmentation errors are mainly caused by large gaps between words. As a result, the most common substitution for a line break is the space character \( \_ \in \Sigma \) in the merged line. Hence, we allow to interpret a line break as space character and the other way around. This is achieved by allowing successive split operations at spaces and merge operations between lines to adjust \( H \):

- split operation: One line \( h = H_y \) with the space character \( h_k = \_ \) at position \( k \) can be split into two lines \( a = (h_1, \ldots, h_{k-1}) \) and \( b = (h_{k+1}, \ldots, h_n) \).
- merge operation: Two subsequent lines \( a = H_y \) and \( b = H_{y+1} \) can be merged into one line \( (a_1, \ldots, a_{|a|}, b_1, \ldots, b_{|b|}) \).

We define the space of partition functions

\[
\Psi := \{ \Phi : (\Sigma^*)^* \rightarrow (\Sigma^*)^* \} \tag{9}
\]

with \( \Phi \) as a composition of split and merge operations. We change (4) by optimizing over all \( \Phi \) that minimizes the LD:

\[
LD^S(H, G) := \min_{\Phi \in \Psi} LD(\Phi(H), G). \tag{10}
\]

and get \( CER^S(H, G) := \frac{LD^S(H,G)}{|G|} \). For the example in (8) and the optimal \( \Phi^* \) we get

\[
H' = \Phi^*(H) = ("Kainz Josina", "Led.");
\]

which leads to \( CER^S(H, G) = CER(H', G) = 0 \).

It has to be mentioned, that \( |H| \neq |\Phi(H)| \) is possible. Furthermore, for an optimal \( \Phi^* \) there is no text line \( H_y \in H^*(H) \) in \( U \) which contains spaces, because a splitting of \( H_y \) at this spaces would result in a lower LD.
Fig. 1. Example of a table with column-wise sort of text lines. Two common LA errors are missing baselines (see $G_2$, $G_3$) and erroneously merging of text lines (see $(G_2, G_3) \rightarrow H_2$). Also the reading order can cause errors: In the HYP, the first two columns are merged together, so that the transcription “Led.” of $G_3$ is ordered before $H_3$, but after $G_2$. Dependent on the configuration, these errors influence the measure (see Figure 2).

Fig. 2. Comparison of Measures. The error rates are calculated from the transcripts and polygons shown in Figure 1. For CER and WER we can define precision (Prec) and recall (Rec) similar to the measures of BOW (see 11,12). In this example $WER$, $WER^G$ result even in the same precision and recall values. Whereas $WER$ finds a correct assignment between $H_0$ and $G_0$, $WER^R$ and $WER^G$ avoid this either by the forced reading order or by the comparison of the corresponding baselines. If we allow segmentation errors, $WER^S$ correctly assigns $H_2$ to $G_2$ and $G_3$.

D. Combination of Measure Modifications

The equations (5), (7) and (10) are defined as single modifications of (4), whereby in many scenarios a combination of these modifications is reasonable: For example to measure the quality of a text extraction method, the semantic meaning is important, which leads to the reading order restriction combined with the option to change the segmentation. We will denote combinations of configurations by adding all modification letters to the superscript (in the previous example: $CER^{R,S}$). Having 3 modifications we can choose between $2^3 = 8$ configuration-dependent CER measures.

Besides the possibility to evaluate the quality of an HTR engine under different restrictions, a meaningful comparison of the results for different measure configurations allows for an examination of the categories of the main errors of this system.

E. From CER over WER to BOW

The WER can be determined based on the CER methodology introduced in this chapter. If $\Sigma$ is not chosen as alphabet of characters but of words instead, everything in Section II holds and the CER becomes the WER. Hence, the WER with all different configurations can be calculated.

There is no general definition of how to transform a sequence of character into a sequence of words. For example, the sequence “it’s” could be divided into one, two or three words. Since in the most cases the user has his own idea of “words”, we provide a simple interface to integrate own word tokenizers. A basic tokenizer, that splits a character sequence at spaces, is implemented as default. For Figure 2 this tokenizer is used.

At first glance CER does not have much in common with BOW. However, by successively changing the configurations, we can close the gap between these measures:

\[
CER^R \leftrightarrow WER^R \leftrightarrow WER \leftrightarrow WER^S \leftrightarrow BOW
\]

So far, it is not obvious, why $WER^S \leftrightarrow BOW$ is reasonable. For the WER calculation we do not only count the manipulations insertion, deletion and substitution, we also count the number of correctly assigned characters/words (COR). For the BOW measure the false positive (FP), the false negative (FN) and the true positive (TP) words are counted. So we can define precision and recall for WER and CER with similar counts used in BOW:

\[
\text{Prec} := \frac{\text{COR}}{|\text{HYP}|} \leq \frac{TP}{TP + FP} = \frac{TP}{|\text{HYP}|} \quad (11)
\]

\[
\text{Rec} := \frac{\text{COR}}{|\text{GT}|} \leq \frac{TP}{TP + FN} = \frac{TP}{|\text{GT}|}, \quad (12)
\]

whereas $|\text{GT}|$ and $|\text{HYP}|$ are the number of characters/words in GT and HYP. Note that in Figure 2 for precision and recall for $WER^S$ and $BOW$ are equal, even with additional geometric restrictions. Since $WER^S$ is constructed to minimize the LD, which only implicitly maximizes COR, the inequality is obvious. But if the lines of G or H are single words, it follows equality and we closed the gap between $WER^S$ and $BOW$.

III. ALGORITHM DESCRIPTION

In this section the implementation details for four out of the eight possible measure configurations, namely

2Interface: https://github.com/Transkribus/TranskribusInterfaces/blob/master/src/main/java/eu/transkribus/interfaces/ITokenizer.java
LD, \(LD^R\), \(LD^{R,G}\), \(LD^{R,S}\), are described. Furthermore, it is discussed if the proposed algorithms result in the minimal LDs – if they solve the minimization problems see (4), (5), (7) and (10) exactly – or not.

Since the set of possible assignment matrices (3) allows for arbitrary line permutations of \(H_{p,y} \in [N]\), its cardinal number exceeds the factorial number \(N!\). Consequently, for practical relevant values of \(N\) the calculation of \(LD\) – the optimization of (2) – becomes intractable and will not be computed exactly. In Sec. III-D a greedy algorithm is introduced to find the (greedy-)optimal assignment matrix \(A \in A\). In the cases of \(LD^R, LD^{R,G}, LD^{R,S}\), the constraint of a fixed reading order, see Sec. III-A allows for the formulation of exact algorithms. In Sec. III-A – III-C these algorithms are introduced and it is proven that they result in global minima for the LDs.

As shown in Section III the LD can be calculated using dynamic programming over subsequences between \(h,g \in \Sigma^*\) which leads to a two-dimensional calculation problem. Because of \(H,G \in (\Sigma^*)^2\), the dynamic programming becomes four-dimensional. We avoid this by flattening \(H,G\) to one dimension in a first step, such that the dynamic programming remains two-dimensional. Therefore, we add the artificial line break character \(\epsilon \in \Sigma\) to the alphabet and get \(\Sigma := \Sigma \cup \{\epsilon\}\). Let

\[ f : (\Sigma^*)^* \to \Sigma^* \]  
(13)

be the invertible flattened function that concatenates the text line and puts \(\epsilon\) before, between and after the lines. For example we obtain

\[ f((a,b),(c,d),(e,f)) = (\epsilon,a,b,\epsilon,c,d,\epsilon,e,f,\epsilon). \]

Finally, the flattened hypothesis and ground truth lines are defined as \(h := f(H), g := f(G)\) with \(h,g \in \Sigma^*\).

In the next sections configure-dependent equations to calculate the LDs for the different restrictions are proposed.

### A. Exact Calculation of \(LD^R(H,G)\)

We use the recursion defined in (2) and expand it to calculate the LD across text lines for the flattened \(h,g\). For that purpose, we expand (1) to

\[ \delta^R_{i,j} := \begin{cases} 0 & \text{if } h_j = g_j \\ 1 & \text{if } h_j \neq g_j \land h_i, g_j \in \Sigma \\ \infty & \text{else} \end{cases} \]  
(14)

This adaptation will prevent substitutions of usual characters by line break characters and vice versa. Consequently, only line breaks can be mapped to each other. Hence, this enforces a direct comparison of entire text lines instead of parts of text lines. Let \(b_h \subseteq [|h|][|h|+1]\) be the tuple of line break positions in \(h\), whereas \(b_g := (b_h)_y \subseteq [|h|]\) is the index of the \(y\)-th line break in \(h\). The tuple \(b_h\) is defined in the same manner. For simplification we use the notation of the cross product of sets for tuples:

\[ (i,j) \in b_h \times b_g \iff i \in b_h \land j \in b_g. \]

For index pairs \((i,j) \in b_h \times b_g\) which represent line breaks at \(i = b_h^x\) and \(j = b_g^y\), we modify the distance calculation in (2) to allow for the deletion and insertion of lines

\[ \Delta^R_{i,j} = \min \left\{ \frac{\Delta^R_{i-1,j-1} + |H_{i-1,j-1}|}{|y|} \text{ if } y \geq 2 \right\} \]  
(15)

for other index pairs \((i,j) \in ([|h|] \times [|g|]) \setminus (b_h \times b_g)\) we set

\[ \Delta^R_{i,j} = \min \left\{ \frac{\Delta^R_{i-1,j-1} + \delta^R_{i,j}}{|y|} \text{ if } h_j \neq \epsilon \right\} \]  
(16)

In the following, we use the term points for index pairs.

### Theorem 1 (Minimal \(LD^R\) calculation)

Let \(h = f(H)\) and \(g = f(G)\) be the flattened sequences. The following equality holds:

\[ LD^R(H,G) = \Delta^R_{|h|,|g|}. \]  
(17)

**Proof.** If for each point \((i,j)\) the minimal predecessor is stored and the final value \(LD^R(h,g) = \Delta^R_{|h|,|g|}\) is calculated, the path leading to the minimal LD can be recursively reconstructed, starting from point \((|h|,|g|)\) until ending in \((0,0)\).

\[ P := (0,0), \ldots, ((h_i,|g|) \in \{\Sigma^2\}^* \text{ the best path. Due to (14) and (15) the path contains all line breaks of } h \text{ and } g. \text{ As shown in Alg. 1 } U,V \text{ and } W \text{ can be obtained from } P. \text{ We use induction over the number of accumulated lines in } H \text{ and } G \text{ (which is } K = |H| + |G|\), to show that (17) holds. \]

For \(K = 0\) we have \(H = G = \emptyset\) and \(LD(H,G) = 0\).

For \(K \geq 1\) with \(H = \emptyset, h = (\epsilon)\) and \(|G| = K \geq 1\), (14) and (15) result into one single path

\[ P := ((0,0), (1, b^1_g), \ldots, (1, b^M_g)) \]

and we can calculate the LD

\[ \Delta^R_{1,b^1_g} = \Delta^R_{1,1} = \Delta^R_{0,0} = 0 \]

\[ \Delta^R_{1,b^M_g} = \Delta^R_{1,b^M_g-1} + |G_{z-1}| = \sum_{j=1}^{z-1} |G_j| \]

\[ LD^R(h,g) = \Delta^R_{|h|,|g|} = \Delta^R_{1,b^M_g+1} = \sum_{j=1}^{M} |G_j|. \]

The same argument can be used for the calculation of \(|H| \geq 1\) and \(G = \emptyset\).

Now, we apply induction over \(K\) for \(|H|,|G| \geq 1\). Let \(H' := H \setminus \{H_{H}\}\) and \(G' := G \setminus \{G_{G}\}\) be tuples of text lines without the last text line. As induction hypothesis we assume \(LD^R(H',G') = (K - 2), LD^R(H,G) = LD^R(H',G) = (K - 1)\) are correctly calculated. We will show that we can calculate \(LD^R(H,G) = \Delta^R_{|h|,|g|}\) using the induction hypothesis.

Let \(h' := f(H')\) and \(g' := f(G')\) be the flattened HYP and GT. Since \(\forall i \in |h'| : h_i = h_i\) it follows (15) will be the same
no matter if we compare with $H$ or $H'$. The same argument holds for $G$ and $G'$.

All paths ending in the point $(|h| - 1, |g| - 1)$ contain
\[
\left( b_{|h| - 1}^h b_{|g| - 1}^g \right) = \left( h^l, g^l \right). \]
So we separately calculate the LD for both parts, which is
\[
\Delta_{|h| - 1, |g| - 1}^R = LD_{|h|}^R(H', G') + LD_{|g|}^R(H_{|h|}, G_{|g|}).
\]
If we set $i = |h| = b_{|h| + 1}^h, j = |g| = b_{|g| + 1}^g$ in (15) and use
$b_{|h|}^h = b_{|h| + 1}^h = |h'|$ and $b_{|g|}^g = b_{|g| + 1}^g = |g'|$ we get
\[
\Delta_{|h|, |g|}^R = \min \left\{ \begin{array}{c}
\Delta_{|h| - 1, |g| - 1}^R \\
\Delta_{|h|, |g|}^R + |H_{|h|}| \\
\Delta_{|h|, |g|}^R + |G_{|g|}| \end{array} \right\}
\]
\[
= \min \left\{ \begin{array}{c}
LD_{|h|}^R(H', G') + LD_{|g|}^R(H_{|h|}, G_{|g|}) \\
LD_{|h|}^R(H', G') + LD_{|g|}^R(H, G_{|g|}) \\
LD_{|h|}^R(H, G') + LD_{|g|}^R(H, G_{|g|}) \end{array} \right\}
\]
Each row indicates how $U, V$, and $W$ are expanded over the recursion: When the first row is the minimum this leads to $(N, M) \in W$, whereby when the second (third) row is the minimum we have $N \in U$ (or $M \in V$). So $LD_{|h|}^R(H, G) = \Delta_{|h|, |g|}^R$ is the minimum of these three sub problems with additional costs as defined in (4).

\[\square\]

**Algorithm 1: SplitBestPath**

| input: $P, b_h, b_g$ |
| output: $U, V, W$ |

1. $U, V, W \leftarrow \emptyset$
2. $p \leftarrow P_2$, $P_1 = (0, 0)$ is not of interest
3. for $i = 3, \ldots, |P|$ do
   4. $q \leftarrow P_i$
   5. if $q_x \in b_h$ then
      6. $x \leftarrow \text{index}(b_h, q_x)$
      7. $x' \leftarrow \text{index}(b_h, q_x)$
      8. $y \leftarrow \text{index}(b_g, q_x)$
      9. $y' \leftarrow \text{index}(b_g, q_x)$
   10. if $y < y'$ then
      11. if $x < x'$ then
         12. $W \leftarrow W \cup \{(y, x)\}$
      13. else
         14. $U \leftarrow U \cup \{y\}$
      15. else
         16. $V \leftarrow V \cup \{x\}$
   17. $p \leftarrow q$ %end point is the new start
18. return $U, V, W$

The calculation of $\Delta_{|h|, |g|}^R$ can be formulated as shortest path problem. Therefore, we search the shortest path from point $(0, 0)$ to $(i, j)$, which indicates the minimal cost to map $h_{1:i}$ to $g_{1:j}$. For $(i, j) = (|h|, |g|)$ we obtain $LD_{|h|}^R(H, G) = \Delta_{|h|, |g|}^R$. Since at each point we calculate the minimum over other points with additional non-negative costs, we can use the Dijkstra Algorithm to solve this problem [7]. Especially for a low CER this algorithm can skip the calculation of many points $(i, j) \in [|h| \times [|g|]$. The implementation is done in Java and freely available on GitHub under the Apache License.

**B. Restricting by Geometric Position**

As mentioned in Section II-B it is reasonable to allow $(y, x) \in W$ only if $H_y, G_x$ are geometrically close to each other ($H_y \in N(G_x)$). To define the neighborhood of $G_x$ we use a method that compares the so-called baselines of the text lines. This is a common measure to evaluate the performance of a layout analysis result [3]. We call the tuple of two-dimensional points $B = (B_1, \ldots, B_{|B|}) \in \mathbb{P}^2 = (\mathbb{N}^2)^*$ a baseline. We define
\[
B^H = (b_1^H, \ldots, b_N^H) \in \mathbb{P}^N
\]
as tuple of polygons corresponding to $H$ and let $B^G$ be defined in the same manner for $G$. From [3 Section III A. 3]) we use the Coverage Function $COV : \mathbb{P} \times \mathbb{P} \times \mathbb{R} \rightarrow [0, 1] \subseteq \mathbb{R}$, which calculates the overlapping between two baselines for a given tolerance value $t : \mathbb{P}^2 \times N \rightarrow \mathbb{R}$ is dependent on the geometric position of all ground truth baselines and the index of the ground truth baseline of interest (cf. [3 Section III A. 2]).

We set
\[
\mathcal{N}(G_x) := \{H_y \in H \mid COV(B_y^H, B_x^G, t(B^G, x)) > 0.0\},
\]
which implicitly restricts the set of valid assignment matrices in (5). Indeed, setting baselines to be close if they have any connection is probably a very soft restriction, but reasonable to avoid erroneously non-assignments for close $H_y$ and $G_x$.

We modify (15) with $i = b_k^h$ and $j = b_k^g$ by
\[
\Delta_{i,j}^L = \min \left\{ \begin{array}{c}
\Delta_{i-1,j-1}^L + |H_y| \quad \text{if } H_y \in \mathcal{N}(G_x-1) \\
\Delta_{i-1,j}^L + |G_x| \quad \text{if } y \geq 2 \\
\Delta_{i,j-1}^L + |G_x| \quad \text{if } x \geq 2 \\
\end{array} \right\}.
\]

**Theorem 2 (Minimal LD_{R,G} calculation).** Let $h = f(H)$ and $g = f(G)$ the flattened sequences. The equation
\[
LD_{R,G}(H, G) = \Delta_{|h|, |g|}^R
\]
holds.

**Proof.** We use Theorem 1 to prove that the additional constrained described in (6) are fulfilled by the changes between (15) and (18). Let $A_{y,x} = 1$ then $H_y \in \mathcal{N}(G_x)$ have to be shown. From $A_{y,x} = 1$ it follows $(y, x) \in W$. But in the proof of Theorem 1 it is shown that $(y, x) \in W$ can only be achieved if in (18) (and 15) the minimum is reached in the first row. This is only possible, if $H_y \in \mathcal{N}(G_x)$.

**C. Non-Penalizing Segmentation Error**

If we allow $\Phi \in \Psi$ to be applied to $H$, we have to modify the LD calculation at some positions. As argued in Section II we allow to map $\psi$ to $\omega$ without costs and vice versa. We

3https://github.com/CITlabRostock/CITlabErrorRate
define $\overline{b}_h$ as expansion of $b_h$ by also containing the positions of the space character $\in \Sigma$. We modify (14) by

$$
\delta_{i,j}^{R,S} := \begin{cases} 
0 & \text{if } h_i = g_j, \\
1 & \text{if } h_i \neq g_j \land h_i, g_j \in \Sigma \setminus \{\cdot\}, \\
0 & \text{if } h_i = \cdot \land g_j = \cdot, \\
\infty & \text{else}
\end{cases} 
$$ (19)

and (15) in points $(i, j)$ with $i = \overline{b}_h^y$ and $j = b_g^x$ by

$$
\Delta_{i,j}^{R,S} = \min \left\{ \begin{array}{l}
\Delta_{i-1,j-1}^{R,S} + \delta_{i,j}^{R,S} \quad \text{if } y \geq 2 \\
\Delta_{i,j-1}^{R,S} + |G_{x-1}| \quad \text{if } x \geq 2 \\
\Delta_{i-1,j}^{R,S} + \delta_{i-j}^{R,S} - 1 \quad \text{if } y \geq 2 \\
\Delta_{i-1,j}^{R,S} + \delta_{i-j}^{R,S} - 1 \quad \text{if } x \geq 2
\end{array} \right\}, 
$$ (20)

which also allows to skip single words. This leads to the updated Algorithm 2, which implicitly returns the best segmentation $H^\prime := \Phi(H)$.

**Algorithm 2: SplitBestPathWithSegmentation**

**Input:** $P, b_h, h$

**Output:** $U, V, W, H^\prime$

1. $U, V, W \leftarrow \emptyset$
2. $H^\prime \leftarrow []$
3. $p \leftarrow P_2$, $%P_1 = (0, 0)$ is not of interest
4. for $i = 3, \ldots, |P|$
do 5. $q \leftarrow P_i$
6. if $q_x \in b_h$ then
7. $x \leftarrow \text{index}(b_h, p_3)$, $%x$-th $\in g$
8. $x^\prime \leftarrow \text{index}(b_h, q_x)$, $%x^\prime$-th $\in g$
9. if $p_3 < q_x$ then
10. $h^\prime \leftarrow b_{p_3+1}q_x-1$,
11. $h^\prime \leftarrow \text{replace}(h^\prime; \cdot; \cdot)$
12. $H^\prime.\text{append}(h^\prime)$
13. if $x < x^\prime$ then
14. $W \leftarrow W \cup \{(H^\prime), x\}$, $%H_y$ maps $G_x$
15. else
16. $U \leftarrow U \cup \{(H^\prime)\}$
17. $V \leftarrow V \cup \{x\}$
18. else
19. $p \leftarrow q$, $%\text{end point is the new start}$

return $U, V, W, H^\prime$

**Theorem 3 (Minimal $LD^{R,S}$ calculation).** Let

$$
\Phi^* = \arg\min_{\Phi \in \Psi} LD^R(\Phi(H), G)
$$

be the best partition minimizing (10). For the $LD$ calculated by (19) and (20)

$$
LD^R(H^*, G) = LD^{R,S}(H, G) := \Delta_{|h|,|g|}^{R,S}
$$

holds. Algorithm returns the best partition $H^* = \Phi^*(H)$.

**Proof.** Clearly, the inequality $LD^R(H^*, G) \leq LD^{R,S}(H, G)$ holds due to the optimality of $H^*$. To show $LD^R(H^*, G) \geq LD^{R,S}(H, G)$, let $h^* = f(H^*)$ and $h = f(H)$ be the flattened sequences. Let $P^*$ be the best path in $LD^{R,S}(H, G)$ with the same cost. Since $h^*$ and $h$ can only differ in $i \in b_h = b_{h^*}$, we only have to show that (15) is equal to (20) in points $(i, j) \in P^*$ with $i = b_{h^*}$. For $j = b_{h^*}^x$, the equations only differ in the path which deletes $H^*_{y_{i}}$ with $i = b_{h^*}^y = b_{h^*}^x$. Because the minimal $|H^*_{y_{i}}|$ is achieved if $H^*_{y_{i}}$ contains no spaces, we know $|H^*_{y_{i}}| = b_{h^*} - b_{h^*} - 1$, so for $j = b_{h^*}^x$ the equations are equal.

For $j \neq b_{h^*}$, (14) and (19) only differ in $\cdot \in g$, but for both possible values $h_i \in \{\cdot, \cdot\}$ we get $\delta_{i,j}^{R,S} = \delta_{i,j}^{R,S}$, so they are equal.

Since the number of possible permutations of the text lines is too large, to exactly calculate the minimal LD, a heuristic will be defined to find the best map between $H$ and $G$. Therefore, (15) is changed at positions $i = b_{h}^y$ and $j = b_{g}^x$ by allowing to ‘jump’ between hypothesis lines:

$$
\Delta_{i,j} = \min \left\{ \Delta_{i-1,j-1} + \delta_{i,j}^{R,S} \right\}. 
$$ (21)

This allows the algorithm to find the optimal $H_y$ for each $G_x$. Due to these jumps Alg. 2 can now return tuples in $W$ having the same value in the first component. This leads to $|A_x| > 1$ and $A \neq A$. The idea for the greedy Alg. 3 is to assign $H_x$ to $G_y$, which minimizes

$$
\arg\min_{H_y \in H} CER(H_y, G_x)
$$

Thus, the algorithm “locally” finds the minimal CER for each $G_x$. The Hyp line with higher CER stays in the set of unmatched lines as well as GT lines, that were not mapped by Alg. 3. On these subsets the algorithm is applied recursively. The number of recursive calls is bounded by $|G|$, because (21) does not allow to skip $G_x$ and at least the first component of $L$ in Alg. 3 Line 7 leads to a reduction of $H$ and $G$. In practice, the recursion depth is between 1 and 4, whereas $G$ is reduced very fast over the depth.

**IV. CONCLUSION AND FUTURE WORKS**

We have introduced a measure to evaluate an end-to-end text recognition system. Depending on its configuration it considers the reading order, segmentation errors and the geometric position. So it closes the gap between a raw character error rate (which so far was only properly defined on text line level) and bag-of-word (which is a retrieval measure on words, that mostly takes the geometric position into account).

Further research can be done to close the gap towards key word spotting (KWS) measures like mean average precision (mAP) or general average precision (gAP).
Algorithm 3: greedy_LD

\[
\begin{align*}
\text{input} & : \text{HYP: } H \\
\text{input} & : \text{GT: } G \\
\text{output:} & \text{greedy minimal LD: } LD(H, G) \\
\text{1} & \text{if } G = \emptyset \text{ then} \\
& \quad \text{return } \sum_{y \in H} |H_y| \\
\text{2} & \text{if } H = \emptyset \text{ then} \\
& \quad \text{return } \sum_{x \in G} |G_x| \\
\text{3} & P \leftarrow \text{runDynProg}(h, g) \\
\text{4} & U, V, W \leftarrow \text{SplitBestPath}(P, b_h, b_g) \quad \text{%see Alg. [1]} \\
\text{5} & L \leftarrow \text{sort}([W; (y, x) :: CER(H_y, G_x)]) \quad \text{%returns array with entries (y, x) sorted by CER} \\
\text{6} & D \leftarrow 0 \\
\text{for } k \leftarrow 1 \text{ to } |L| \text{ do} \\
& \quad (y, x) \leftarrow L[k] \\
\text{7} & \text{if } H_y \in H \text{ then} \\
& \quad H \leftarrow H \cup \{H_y\} \\
\text{8} & \quad G \leftarrow G \cup \{G_x\} \\
\text{9} & \quad D \leftarrow D + LD(H_y, G_x) \\
\text{10} & \text{return } D + \text{greedy_LD}(H, G)
\end{align*}
\]

ACKNOWLEDGMENT

This work was partially funded by the European Union’s Horizon 2020 research and innovation programme under grant agreement No 674943 (READ – Recognition and Enrichment of Archival Documents).

REFERENCES

[1] J. Sivic and A. Zisserman, “Efficient visual search of videos cast as text retrieval,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 31, no. 4, pp. 591–606, 4 2009.
[2] U.-V. Marti and H. Bunke, “Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system,” in Hidden Markov models, applications in computer vision. World Scientific, 2001, pp. 65–90.
[3] T. Grütting, R. Labahn, M. Diem, F. Kleber, and S. Fiel, “READ-BAD: A new dataset and evaluation scheme for baseline detection in archival documents,” CoRR, vol. abs/1705.03311, 2017. [Online]. Available: [http://arxiv.org/abs/1705.03311]
[4] M. Murdock, S. Reid, B. Hamilton, and J. Reese, “ICDAR 2015 competition on text line detection in historical documents,” in Proceedings of the International Conference on Document Analysis and Recognition, ICDAR, vol. 2015-November. IEEE, 8 2015, pp. 1171–1175.
[5] S. Pletschacher and A. Antonacopoulos, “The page (page analysis and ground-truth elements) format framework,” in 2010 20th International Conference on Pattern Recognition, 8 2010, pp. 257–260.
[6] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” in Soviet physics doklady, vol. 10, no. 8, 1966, pp. 707–710.
[7] E. W. Dijkstra, “A note on two problems in connexion with graphs,” Numer. Math., vol. 1, no. 1, pp. 269–271, Dec. 1959. [Online]. Available: [http://dx.doi.org/10.1007/BF01386390]