An open-source OCR evaluation tool

Rafael C. Carrasco
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante (Spain)

ABSTRACT

This paper describes an open-source tool which computes statistics of the differences between a reference text an the output of an OCR engine. It also facilitates the spotting of mismatches by generating an aligned bitext where the differences are highlighted and cross linked.

The tool accepts a variety of input formats (both for the reference and the OCR) and can also be also used to compare the output of two different OCR engines. Some considerations on the criteria to compare the textual content of two files, at character and word level, are also discussed here.

Keywords

OCR evaluation, text transcription, open-source code

1. INTRODUCTION

The digitization of textual content usually involves four complementary steps: image capture, transcription, dissemination and long-term preservation. Optical character recognition (OCR) deals with the transformation of a digital image —for example, a digital picture of a book page in JPG format— into a digital document whose text can be easily processed by a computer —such as a text file or searchable PDF. OCR enriches the digital text since it makes the content indexable, and facilitates its transformation into other formats for dissemination or preservation.

The existing OCR engines provide high-quality results for modern printed text. There is, however, a large room for improvement for the efficacy of OCR on historical text or manuscripts. The current technology, when applied to historical documents —for example, 16th century books— often leads to text where most of the words are recognized wrong. The reasons for such a low performance are multiple: old fonts which are not adequately interpreted by modern OCR devices; bad quality of the original image (irregular spacing between characters, stains, warped text, variable contrast, transparency); complex layout which makes it difficult to find the correct reading order; ancient vocabulary or orthography that poses additional difficulties to the disambiguation of alternative interpretations of the content.

DATeCH 2014, Madrid, Spain
ACM 978-1-4503-2588-2/14/05
http://dx.doi.org/10.1145/2595188.2595221

The insufficient performance of OCR calls for the definition of objective quality measures[9, 13] which allow the users to compare different devices and also the researchers and developers to guide their efforts. It also requires tools to evaluate automatically the quality of the transcription. Some of the existing tools measure the performance in the layout analysis[6, 2], since the correct segmentation of an image is essential to obtain a transcription with the correct reading order[3] —for example, newspaper pages often include multiple columns, headings, caption, etc. Other tools are however oriented to evaluate the accuracy in the interpretation of the content (the printed characters and words). This type of evaluation compares the output with a reference which contains a highly accurate transcription (or ground truth) of the source document[12]. The creation of such ground truth is expensive but usually a limited size one is enough to obtain significant numbers provided that it contains a representative sample of the collection.

Unfortunately, many of the available tools for the evaluation of the accuracy of a transcription are proprietary[1] and, therefore, difficult to adapt to particular environment —for example, a specific output format. The INL word evaluation tool[4] accepts input text in PAGE[8] and and ALTO formats and provides precision, recall, and error rate in dictionary words. The PRImA OCR evaluation tool —used for the 2013 ICDAR Competition on Historical Book Recognition[1] —, implements standard word and character accuracy measures[10] for PAGE and plain text inputs complemented with a bag-of-words method which is independent from the reading order.

The open-source `hocr-eval` tool[2] can be used to compute the distance (number of OCR errors) between files in hOCR format, Tesseract's output in HTML format[11]. Our `ocrevaluation` tool[3] is also an open-source code in Java which computes word and character errors rates for a variety of formats including plain text, PAGE[8], ALTO[4] and hOCR[5]. Furthermore, it aligns the two input texts and highlights the differences between them. In contrast to general-

[1]See, for instance, those listed at `http://www.digitisation.eu/tools/demonstrator-platform`
[2]`https://github.com/CUSAT/hocr-tools`
[3]`http://www.digitisation.eu/training/succeed-training-materials/ocrevaluation`
[4]`www.loc.gov/standards/alto`
[5]For a specification, see `http://docs.google.com/View?docid=dfxcv4vc_67g844kf`.

purpose text-comparison tools —such as the Unix commands diff or wdiff—, the tool is Unicode aware and allows for the custom definition of equivalences between glyphs —such as medieval[6] and modern characters or Unicode compatibility tables[7].

2. MEASURING OCR ACCURACY

The output of OCR engines often contains a number of mistakes such as misspelled words or spurious characters. In order to obtain a measure which is independent of the text size, the number of mistakes is usually normalized to the length of the expected content (the ground truth text). The quotient between the number of mistakes and the text length is known as *error rate* and it is usually computed at two different levels: *character error rate* (CER) and *word error rate* (WER).

It is worth to be remarked that word error rates are usually higher than character error rates[13]. For instance, a relatively small number of wrong characters, such as a 10% error rate, means that about one half of six-letter words will contain at least one wrong character (under the naive assumption that the distribution of errors among characters is homogeneous). Nevertheless, a good correlation between both measures can be expected (except, perhaps, for very high error rates) when both measures are used to compare different OCR engines.

Experiments suggest that humans are relatively intolerant to errors in the following sense: a word accuracy below 85% leads to a lower productivity if manual correction is applied to the output compared to that of typing entirely from scratch. For example, it has been claimed[5] that good OCR accuracy means about 98-99% accurate, while an accuracy below 90% will be considered poor.

The estimation of error rates requires a reference text, a digital document which has been manually produced to represent, as accurately as possible, the textual content of the original source. Such transcriptions are often called ground truth (a term borrowed from cartography), since they map the source to the true textual content. Although the true content seems to be a clear concept, its creation often needs to deal with some subtle decisions. For example, should ligatures be represented as a single Unicode character or as two independent characters? Or, will blurred and distorted letters be transcribed as the one pretended to be or as they actually look like? There is no final answer to such type of questions, since only the source image contains all the original information and a meticulous creation of ground truth can demand an effort too high.

2.1 Types of errors

The output of an OCR engine usually contains different types of errors:

- Misspelled or confused characters (substitutions).
- Spurious symbols (insertions).
- Lost or missing text (deletions).

For example, the following line of text

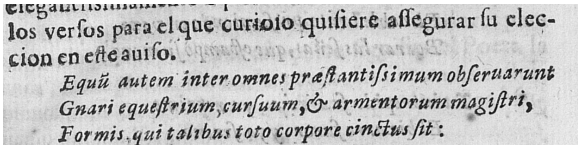

los verſos para el que curioſo quiſiere aſſegurar ſu eleccion en eſte auiſo.
Equũ autem inter omnes præſtantiſsimum obſeruarunt
Gnari equeſtrium, curſuum, & armentorum magiſtri,
Formis, qui talibus toto corpore cinctus ſit:

Figure 1: An image with different ligatures (long s + i, long s + long s, and long s + t) and a character et.

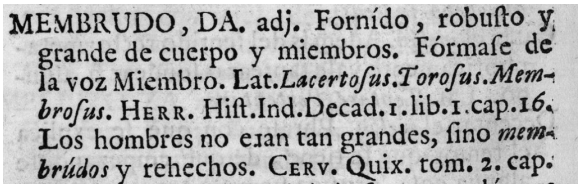
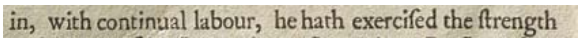
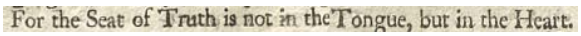

MEMBRUDO, DA. adj. Fornído, robuſto y grande de cuerpo y miembros. Fórmaſe de la voz Miembro. Lat. *Lacertoſus. Toroſus. Membroſus.* HERR. Hiſt. Ind. Decad. 1. lib. 1. cap. 16. Los hombres no eran tan grandes, ſino *membrúdos* y rehechos. CERV. Quix. tom. 2. cap.

Figure 2: A fragment with ligatures and an inverted r (fifth line) which looks like an i.


in, with continual labour, he hath exerciſed the ſtrength

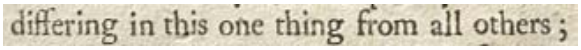
has been read by the OCR engine as "be hath exerciled the ftrength" instead of the original "he hath exercifed the ftrength". Clearly, there are three misspelled characters in the produced reading: the h in word "bh" has been misinterpreted as a b, and two long s have been read as I and f characters respectively.

However, the following line


For the Seat of Truth is not in the Tongue, but in the Heart.

was read as "For the Seat of Truth is not m theTongue, but in the Heart.". In this example, the word "in" has been interpreted as "m" which can be understood as an "i" which has disappeared followed by an "n" confused with an "m". Also the space between "the" and "Tongue" has been lost.

Finally, the text segment


differing in this one thing from all others;

was read as "differing in this one thing from all others';" where an spurious apostrophe has appeared, apparently due to some dirt in the image.

In some applications, for example, those dealing with text typed by humans, swapping two characters is a common error. However, anagrams are not a natural mistake in OCR, and therefore we will not consider them as a specific type of error but rather as equivalent to the concurrence of two basic operations: a deletion followed by an insertion. For example, the word "nuclear" can be replaced with "unclear" if the leading "n" is removed and then a new "n" is inserted before "clear".

2.2 Computing error rates

Computing an error rate is not so simple when the number of mistakes grows. For instance, one could argue that the character error rate between the expected output "ernest" and "nester" is 6 since no character in the reference appears in "nester" at the required position. However, two identical subsequences "nest" can be aligned in both words. A subsequence of a word is any string of characters that can be

[6]Medieval Unicode Font Initiative, `http://www.mufi.info/`.

[7]`http://unicode.org/reports/tr15/#Canon_Compat_Equivalence`

obtained by just removing some of the original ones (no sorting allowed since the order of letters is essential for correct spelling).

Clearly, there is more similarity between this pair of words than that conveyed by the suggested 100% CER: indeed it is enough to remove 2 extra characters ("er") at the beginning of the the first word and then insert the two trailing characters ("er") in order to obtain the second word. With this 4 operations, the CER becomes about 67%. In contrast, the alignment of just both "er" subsequences would require 8 operations and, therefore, lead to a CER of 133%.

Therefore, the CER is computed with the minimum number of operations required to transform the reference text into the output (this number which is known as the Levenshtein distance[7]). Although finding the optimal set of basic operations is not trivial for lengthy and noisy outputs, there there are computational procedures which can evaluate this number efficiently[14].

With these numbers, a standard definition of the *word error rate* is:

$$\mathrm{WER} = \frac{i_w + s_w + d_w}{n_w}$$

where n_w is the number of words in the reference text, s_w is the number of words substituted, d_w the number of words deleted and i_w the number of words inserted required to transform the output text into the target. Among all possible transformations, that one minimizing the sum $i_w + s_w + d_w$ is chosen and then the number c_w of correct words is $c_w = n_w - d_w - s_w$.

Similarly, the *character error rate* is defined as:

$$\mathrm{CER} = \frac{i + s + d}{n}$$

which employs the total number n of characters and the minimal number of character insertions i, substitutions s an deletions d required to transform the reference text into the OCR output.

As seen before, the number of mistakes can be larger than the length of the reference text and lead to rates large than 100%. Thus, the number of errors is sometimes divided by the sum $i+s+d+c$ of the number of edit operations $i+s+d$ plus the number c of correct symbols, which is always larger than the numerator.

2.2.1 White space

Blank or white space plays an important role in any text, since it separates words (e.g., it is quite different to read "mangoes" instead of "man goes"), and it must be therefore considered while computing the character error rate. In order to understand a text, it is however not so relevant the length of the white space. Therefore, contiguous spaces are often considered to be equivalent to a single one, that is, the CER between "were␣␣wolf" and "werewolf" is 0.125.

However, white space characters are not like the others, since allowing the alignment between texts to replace a white space with a printable character may lead to weird results. For example, when comparing "bad man" and "batman", the deletion of d and the substitution of the white space with t has identical Levenshtein cost (two operations) than the deletion of the white space plus the replacement of the d with a t (also two operations); clearly, the second option is a more natural transformation. This undesired result can be minimized if the substitution of white space with printable characters is not allowed during the computation of the distance and the associated character alignments.

In contrast, the computation of word error rated does not need to take care of white space since the text is preprocessed by a word tokenizer and blanks only matter in this step. Some care must be however taken when defining a word, since the definition may influence word counting. For example, acronyms like "I.B.M." will be split into three different words by many tokenizers making it difficult to retrieve the full word.

2.2.2 Case folding

Clearly, case matters in OCR: for example, "white house" and "White House" probably refer to two different objects. Therefore, the CER when both expressions are compared should not be 0 but rather 0.18 (18%). In the context of information retrieval (whose objective is to find all attestations of a word or expression), case is often neglected so that, for example, capitalized words in titles are equivalent to non-capitalized ones in normal paragraphs. This suggests that the WER, in contrast to CER, should not count as mistakes the substitution of one uppercase letter by the corresponding lowercase one.

2.2.3 Text encoding

There is a large number of alternative encodings for text files (such as ASCII, ISO-8859-9, UTF8, or Windows-1252). Essentially, the encoding maps every character to a unique integer number (its code). Since the number of codes can be large, they are often stored in compact forms.

XML files customarily include the declaration of the encoding in their fist line with the form `<?xml version="1.0" encoding="utf-8"?>`, and this provides a method to detect automatically the character set which was employed to create the file.

Also HTML files often contain meta-information in the header —for example, `<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">`— which identifies what character set was used.

However, these metadata are not compulsory and may be missing in some cases (and they will be always missing in the particular case of plain text files). In all cases where the encoding is not declared in the document, the tool must guess which one has been used. This option will work in most cases (at least for long enough files) but can lead occasionally to unexpected results.

2.3 Unicode

Unicode is a standard to represent text in any writing system. It provides good coverage of modern scripts (describing well over 100.000 glyphs) but often lacks support for old glyphs[8]. For example, Unicode assigns the number 10016 (or 2729 in hexadecimal) to the Maltese cross (✠) but has no code for the triple ligature of long s + long s + i (ffi).

Some consortia, such as the Medieval Unicode Font Initiative, use the so-called Private Use Areas (PUA) to represent characters or glyphs. The PUA are ranges of codes which will remain officially unassigned and whose meaning can be defined by users for a particular context.

Since the production of ground truth may require in some cases the encoding of specific glyphs, it is necessary to define

[8]A glyph is a possible representation of a character or a sequence of characters

```
FB00, 0066 0066, Latin small ligature ff
F50D, 0071 0301 A76B,  ligature q +  acute +  et
FEFF, 0020, byte order mark
...
```

Figure 3: A file storing Unicode equivalences

when a certain output of the OCR engine can be accepted as equivalent to the ground truth character. For example, a q with tilde (q̃) can be considered equivalent to a q in some contexts but equivalent to the word "que" in historical Spanish.

The OCR evaluation tool described in section 3 allows the user to define equivalences between Unicode character sequences. If the difference between the ground truth text and the OCR output can be understood as the substitution of a character or sequence by another equivalent character or sequence, this difference will not be considered a mistake and will not contribute to the error rates.

The equivalences will be stored in a text file which contains two equivalent sequences per line, separated by comma (comments can optionally follow after another comma, as in the CSV files). The characters are described by the Unicode code in hexadecimal notation, as shown in figure 3.

3. THE OCREVALUATION TOOL

The OCR evaluation tool described here[9] allows for the comparison of the reference text with the OCR output and also for the comparison of the output of two different OCR engines or from a single engine when different options are selected.

3.1 Input

The tool creates a report on OCR quality for a particular image using two sources of information, the output of the OCR engine, and the ground truth content of the image. Both inputs are required by the tool and currently the program accepts input files in any of the following formats:

- Plain text (ASCII or Unicode).
- PAGE XML.
- ALTO XML.
- FineReader-10 XML
- hOCR HTML.

3.2 Output

The report generated by the evaluation tool contains:

- The estimated CER (character error rate) and WER (word error rate) for the sample (in several flavors).
- The best alignment between the ground truth text and the OCR text.
- Details statistics on the number of mistakes for every character.

[9] `https://github.com/impactcentre/ocrevalUAtion`

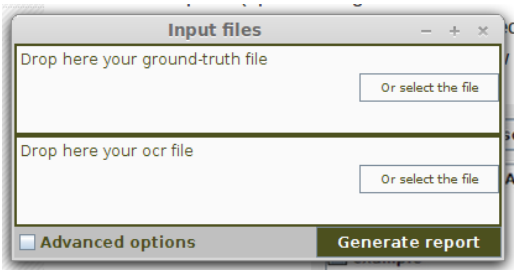


Figure 4: Basic user interface

For example, the output shown in figure 6 has a high word error rate (nearly 70%), partly due to incorrect ordering, as indicated by the lower order-independent word error rate. The alignments are shown in two parallel columns (ground truth appears on the left-hand side and the OCR text on the right-hand side) where:

- Replaced/confused content (substitutions) is colored, and the replacing text is shown when the pointer moves over the highlighted text (the associated segment in the parallel text will be highlighted).
- Spurious content (insertions) is marked with a light background in the OCR text.
- Lost content (deletions) is marked with a light background in the ground truth text.

Finally, a table with all characters in lexicographic order displays the absolute and relative number of mistakes for each one. The hexadecimal code are included to facilitate the identification of those characters which are non-printable (such as blanks) or not properly displayed by the browser.

3.3 Installation and basic usage

The tool can be easily installed following these steps:

1. Download the latest version from the address `http://impact.dlsi.ua.es/ocrevaluation`.
2. Check that the system has Java 6 (or later) installed.
3. Set the file permissions (if using Linux) to allow the execution of the `ocrevaluation.jar` file[10]

Then, a double click on the file (under Windows) or a right-click selecting "open with" and then the application called "Java 6 Runtime" will launch the program and a window like the one displayed in figure 4 will open.

The window has two input areas (the white areas with dark text inside) where the user can drag and drop input files from the computer. Alternatively, the file can be selected using a standard selection window: a click on "Or select the file" will open a standard file selector window and will allow one to navigate through the local file system.

Whenever a file is selected or dropped in the input area, the full name of the file appears. The first input is the ground truth file in any of the supported formats; the second is the OCR output (again, in any of the supported formats).

[10] Typing `chmod a+x ocrevaluation.jar` on your terminal or right clicking on the file, selecting properties and then permissions and clicking on "Allow executing file as program".

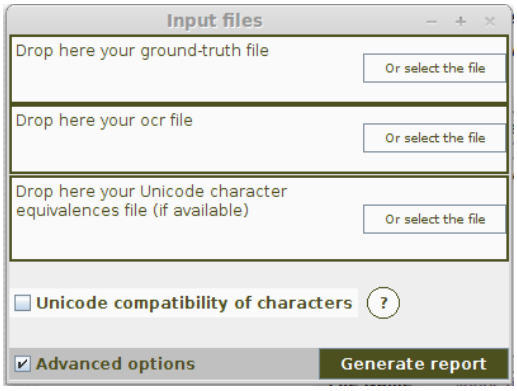


Figure 5: Advanced user interface

Once all the input files has been uploaded, one can lick on the "Generate report" button. If either the required ground truth or the OCR file is missing, then the associated area will be highlighted and the tool will wait till they are all selected. Otherwise, it will open a window to select the name and location of the file which will contain the report in HTML format (by default a file in the same directory as the OCR file and whose name ends with _report.html after the OCR file name).

Finally, the program will open a browser showing the report. If, due to security restrictions, the system does not allow the program to open the browser, the output file can always be examined by clicking on it in a navigator window.

3.4 Advanced usage

3.4.1 *Working with multiple input files*

An accurate evaluation of OCR recommends using a representative sample of pages (showing, for example, the variable page layouts found in different sections of a book). In order to facilitate the global evaluation of multi-page collections, the `ocrevaluation` tool accepts also folders as input and then compares the pairs of files found inside those folders. The procedure is identical to that followed for single files (drag and drop or select a folder): the number of files in both folders must be identical and the files to be compared should have names which include the same identifier (for example, page22_gt.txt and page22_ocr.xml).

3.4.2 *Advanced options window*

If the check-box next to "Advanced options" is marked, then the input window is expanded as shown if fig. 5. The additional input area allows for the upload of a file containing equivalences between Unicode characters, as described in section 2.3. It is also possible to enable the default equivalences defined by the Unicode standard as *compatibility equivalence*[11]: this option will make, for example, equivalent the glyph representing the ligature "ff" and the two character sequence "ff".

3.4.3 *Command line usage*

The tool also allows for command line execution with the following syntax:

```
java -jar ocrevaluation.jar -gt ground_truth_file
      [encoding] -ocr ocr_file [encoding] -d
   output_directory [-r equivalences_file] [-c]
```

where ground_truth_file is the full path to a ground truth file; ocr_file is the full path to the OCR result file; output_directory is the folder where the report will be generated; encoding is the name of the encoding used in the preceding file; equivalences_file is an optional text file describing equivalences between Unicode characters; and the option -c activates the compatibility between Unicode characters.

3.5 Implementation

The evaluation tools builds an edit sequence from two input texts in linear time with respect to the size of the input files and it can therefore be applied to long documents. Typically, it processes thousands of character (several pages) per second on a standard desktop or laptop computer. The input is processed both as a sequence of characters (to compute the CER) and as a sequence of tokens (to evaluate WER).

The linear time cost of the computation is achieved by splitting the input into fixed-length chunks. After one reference and one OCR chunks have been aligned to minimize the Levenshtein distance, only the initial half of the operations in this edit sequence are accepted and appended to a global sequence (initially empty). Then, the text-shifts generated by the operations in the global sequence are computed and the alignment process restarts at that point (where new chunks are obtained).

Finally, the cost associated with the concatenated edit sequence is computed and normalized to produce the error rate. Also, tags are added to the input texts to mark insertions, deletions and substitutions, according to the information provided by the edit sequence.

4. CONCLUSIONS

The `ocrevaluation` open-source software has proven a useful tool to accelerate the evaluation of the performance of different OCR engines upon historical documents. Fed with a limited sample of ground truth pages and the output obtained with the application of different OCR engines, it provides rich statistics to compare their performance.

The code is distributed under the terms of the GNU General Public License (version 2) and can therefore be easily adapted for specific productive environments.

Future work will include the support for additional input formats, the analysis of segmentation errors, and the integration of probabilistic language models in order to obtain indicators of the quality of the output when the creation of a ground truth sample is not viable.

5. REFERENCES

[1] A. Antonacopoulos, C. Clausner, C. Papadopoulos, and S. Pletschacher. Icdar 2013 competition on historical book recognition (hbr 2013). In *2013 12th International Conference on Document Analysis and Recognition, Washington, DC, USA, August 25-28, 2013*, pages 1459–1463. IEEE, 2013.

[2] D. Bridson and A. Antonacopoulos. A geometric approach for accurate and efficient performance evaluation of layout analysis methods. In *19th*

[11] `http://unicode.org/reports/tr15/#Canon_Compat_Equivalence`

CER	25.30
WER	69.75
WER (order independent)	57.14

Difference spotting

00439425.xml	00439425.html
GUZMAN DE ALFARACHE venia cedula de su Mageſtad, en que absolutamente lo mandaſſe, porque asſi se lo suplicavan, y lo embiaron consultado. Aqui di punto y fin à eſtas desgracias; re- matè la quenta con mi mala vida, la que despues gaſtè todo el reſtante della, veràs en la tercera y ultima parte, ſi el cielo me la diere antes de la eterna que todos espe- ramos. Fin de la Segunda Parte. APROBACION. I 396 Nfrascriptus legi Libros del Picaro GuZMAN de ALFARACHE, in quibus nihil reperi quod Catholicæ Fidei adversetur, quare eosdem utiliter imprimi poſſe censeo. Datum Antverpiæ 13. Martij 1677. Aubertus Vanden Eede, Canonicus T. V. L. Librorum Censor Antverpiæ. TABLA	396 GuzMAN DE ALrAnAcniz" 'venia cedula de, fu Magïeficad, en que abfolutamcntc lo' mandan}: ,porque afli fe lo fuplicavan, 'y lo embiaron confultado. Aquí dipunto yfin áefias defgracias ;, rc-|- matè la qucnta con mi mala xridaflla que dcfpucs gaflé todo el refiante della, verás enla tercera y ultima parte, fi el ciclo me la dicrc antes dela eterna que todos cfpc- HNOS. Fin de la Segunda Parte. vv-v ÏAAPRO ÉACION. I Nfrafcríptus legí Libros del Pícaro G uzMAN de A LI-' A n A c H E, in "quibus nihil rcperi quod Catholícac Fídci advcrfetur, quarc cofdem utiliter imprimi poſic Cenfeo. Datum Antverpím 13. Martí} .167 7 '. V . Aubcrtus Vandcn Ecde, Canonícps T. V. L. Librorum Cenfor" Antvcipi-Sfi. i iTABLA

Error rate per character and type

Character	Hex code	Total	Spurious	Confused	Lost	Error rate
	20	112	20	1	6	24.11
"	22	0	2	0	0	Infinity
'	27	0	1	0	0	Infinity
,	2c	9	1	2	0	33.33
-	2d	2	4	0	0	200.00
.	2e	11	3	0	0	27.27

Figure 6: A sample output showing global error rates, the aligned texts and per character statistics.

International Conference on Pattern Recognition (ICPR 2008), December 8-11, 2008, Tampa, Florida, USA, pages 1–4. IEEE, 2008.

[3] C. Clausner, S. Pletschacher, and A. Antonacopoulos. The significance of reading order in document recognition and its evaluation. In *12th International Conference on Document Analysis and Recognition, Washington, DC, USA, August 25-28, 2013*, pages 688–692. IEEE, 2013.

[4] J. de Does. Impact lexica in OCR and IR. `http://www.slideshare.net/impactproject/jesse-de-does`, 2011.

[5] R. Holley. How good can it get? Analysing and improving ocr accuracy in large scale historic newspaper digitisation programs. *D-Lib Magazine*, 15(3/4), 2009. Unpaginated/on-line.

[6] J. Kanai, S. Rice, T. Nartker, and G. Nagy. Automated evaluation of OCR zoning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(1):86–90, January 1995.

[7] V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[8] S. Pletschacher and A. Antonacopoulos. The PAGE (Page Analysis and Ground-truth Elements) format framework. *Proceedings of the 20th International Conference on Pattern Recognition (ICPR2010), Istanbul, Turkey, August 23-26, 2010*, pages 257–260, August 2010.

[9] S. Rice, J. Kanai, and T. Nartker. An evaluation of ocr accuracy'. In *SDAIR93*, page xx, 1993.

[10] S. V. Rice, F. R. Jenkins, and T. A. Nartker. The fifth annual test of ocr accuracy, 1996.

[11] R. Smith. An overview of the Tesseract OCR engine. In *Proc. Ninth Int. Conference on Document Analysis and Recognition (ICDAR)*, pages 629–633, 2007.

[12] N. Stamatopoulos, G. Louloudis, and B. Gatos. A comprehensive evaluation methodology for noisy historical document recognition techniques. In *Proceedings of The Third Workshop on Analytics for Noisy Unstructured Text Data*, AND '09, pages 47–54, New York, NY, USA, 2009. ACM.

[13] S. Tanner, T. Muñoz, and P. H. Ros. Measuring mass text digitization quality and usefulness. *D-Lib Magazine*, 15(7/8), 2009. Unpaginated/on-line.

[14] R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *J. ACM*, 21(1):168–173, 1974.