PRImA Text Evaluation Tool – User Guide

Performance Evaluation for OCR Methods

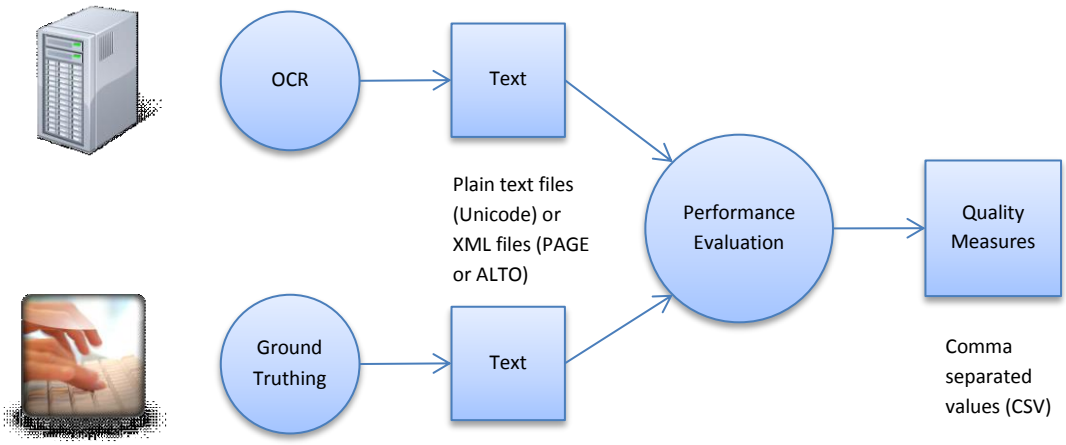
Version 1.4
July 2017

www.primaresearch.org

Introduction

The PRImA OCR Evaluation tool implements the word and character accuracy measures developed by the University of Nevada Las Vegas (UNLV dissertation by S. V. Rice). It has been complemented by a bag-of-words method which is independent from the reading order.

Typical use scenario:



Performance Measures

1. Bag-of-Words Measure

Definition 'Bag of Words' (source: Wikipedia): *The bag-of-words model is a simplifying assumption used in natural language processing and information retrieval. In this model, a text (such as a sentence or a document) is represented as an unordered collection of words, disregarding grammar and even word order.*
The quality of a text (for instance an OCR result) can be evaluated by comparing its Bag of Words against the Bag of Words of the ground truth text.

The evaluation metric is divided in two sections. The first section targets indexing scenarios where only the fact is of interest if a word has been recognised or not, the number of instances of the word is irrelevant. A Bag of Words is then constructed as a set of unique words. The second section of the metric also takes into account the correct number of recognitions of a word (count based measures).

Output values

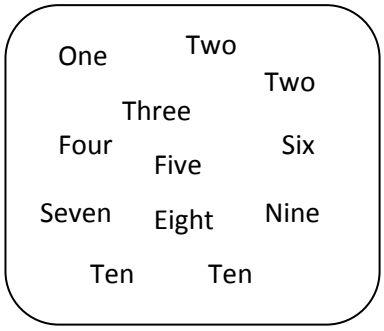
wordIndexMissErrorRate
wordIndexFalseDetectionErrorRate
wordIndexSuccessRate
numberOfUniqueWordsInGroundTruth
numberOfUniqueWordsInResult
wordCountMissErrorRate
wordCountFalseDetectionErrorRate

wordCountSuccessRate
wordCountPrecision
wordCountRecall
wordCountFMeasure
numberOfWordsInGroundTruth
numberOfWordsInResult

Example

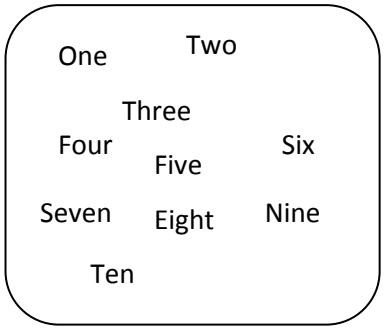
Ground truth:

Bag of Words



(12 words)

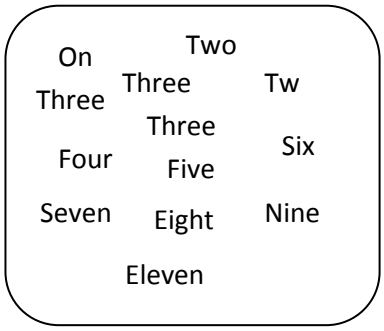
Set of unique words



(10 words)

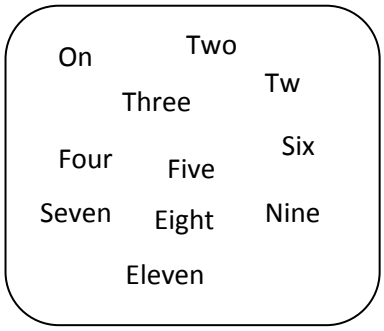
OCR Result:

Bag of Words



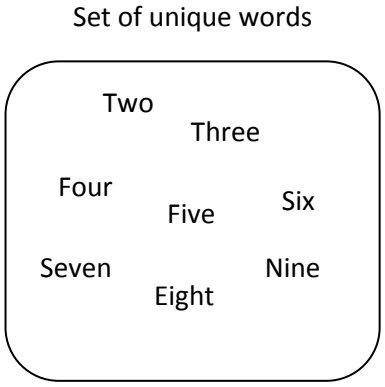
(13 words)

Set of unique words



(11 words)

Intersection:



Index based measures:

Miss = 2 / 10 = 20% error
False detection = 3 / 11 = 27% error
Success rate = 2*(0.58/1.53) = 76% success

Count based measures:

Recall = 8 / 12 = 67%
Precision = 8 / 13 = 62%
F-Measure = 2*(0.41/1.28) = 64% success

Miss = 1 - 7.5/10 = 25% error
False detection = 5 / 13 = 38% error
Success rate = 2*(0.47/1.37) = 68% success

2. Bag-of-Characters Measure

The Bag of Characters measure resembles the Bag of Words measure, but it calculates all values on character level. Currently, multi-part characters (characters that consist of multiple Unicode components) are treated as multiple separate characters.

Output values

characterMissErrorRate
characterFalseDetectionErrorRate
characterSuccessRate
characterPrecision
characterRecall
characterFMeasure
numberOfCharactersInGroundTruth
numberOfCharactersInResult

Statistics

This measure also supports low-level statistics (-csv-stats command line option). Below is an example output table with character statistics.

Ground Truth Item	Ground Truth Unicode	OCR Result Item	OCR Result Unicode	OCR Error	Count
O	004f			none	1
n	006e			miss	1
		e	0065	false detection	4
			0020	false detection	2
t	0074			none	6
w	0077			none	2
o	006f			miss	1
		h	0068	false detection	2
		r	0072	false detection	2
f	0066			miss	1
u	0075			none	1

This means for instance:

- The character “n” was missed once by the OCR, when compared to the ground truth
- The character “e” was found more often by the OCR that it appears in the ground truth (4 times falsely detected)

These statistics are only an approximation. The evaluation method cannot detect if an “x” was missed in one word and falsely detected in another word, for instance. The overall success rate would not be lowered in such a case.

3. Character and Word Accuracy

Character and word accuracy reflect the effort to correct a given text so that it matches a certain expected text. Often the text under test is an OCR result and the expected text is the ground truth. The correction effort is determined by calculating the edit distance it takes to convert one string into another. The PRImA lab implementation is based on Stephen V. Rice’s dissertation ‘Measuring the accuracy of page-reading systems’.

Edit Distance

The edit distance reflects the number of insertions, deletions and substitutions to convert one string into another. A cost function applies error values to the three different correction steps. Depending on the cost function, different algorithms have to be used to calculate the edit distance.

Example cost functions (insertion,deletion,substitution):

(1,1,1) – Insertions, deletions and substitutions equally count as one error.
(1,1,2) – Insertions and deletions count as one, substitutions as two. This is useful for scenarios where a substitution is regarded as one deletion plus one insertion.
(1,0,1) – Deletions are ignored. That way, invented parts of a text are not penalised.

Example:

Expected string: hello
Result string: olloo

Insertions: 1 (h)
Deletions: 1 (o)
Substitutions: 1 (o with e)

Edit distance for (1,1,1): 3
Edit distance for (1,1,2): 4
Edit distance for (1,0,1): 2

Character Accuracy

The character accuracy is the result of the comparison of two strings character by character. Based on the edit distance d the accuracy is defined by:

$$accuracy = \max(0, \frac{m-d}{m})$$

where m is the length of the ground truth string.
The cost function for the edit distance is set to (1,1,1) by default but can be changed to any supported cost function.

Output values:

Name	Range	Description
characterAccuracy	[0.0...1.0]	Character accuracy
charsInGroundTruth	>= 0	Number of characters in the ground truth text
charsInResult	>= 0	Number of characters in the result text

Character Accuracy Statistics

The character accuracy measure supports low-level statistics (-csv-stats option). The statistics are calculated using a heuristic based on Longest Common Substrings (LCS). The ground truth text and the OCR result text are thereby recursively compared and split into three parts each: Left, LCS match, right. The LCS match is reported in the statistics as 'no OCR error'. The left and right parts are further processed using the same approach. This is repeated until the one of the parts is empty (miss or false detection) or they don't match at all (misrecognised).

Example:
Ground truth text:

"*One two two three four five archæology*
six seven eight nine ten ten."

OCR result:

"*On two tw three three three four FIVE archaeology six seven eight nine eleven*"

Statistics output:

Ground Truth Item	Ground Truth Unicode	OCR Result Item	OCR Result Unicode	OCR Error	Count
three four five arch				none	1
two tw				none	1
six seven eight nine				none	1
on				none	1
o		three three		misrecognised	1
ology				none	1
en				none	1
e				miss	1
æ		ae		misrecognised	1
				misrecognised	1
t		elev		misrecognised	1
ten.				miss	1

This is just an approximation and will work better if there are not too many OCR errors. The example shows that even if ground truth and OCR result are quite far apart, useful insights can be found, such as the misrecognition of the ligature ae as separate characters a e.

Flex Character Accuracy

This is a variation of the traditional character accuracy. Because it is based on the edit distance between the ground truth text and the result text, the character accuracy is strongly influenced by the order of paragraphs (or other elements) in the text. For single-column documents this is usually not a problem, but for complex page layouts this measure returns less meaningful results (a low success rate might be due to differences in reading order and not due to bad OCR performance).

The flex character accuracy is also based on the edit distance, but works line-by-line instead of the whole text. Iteratively each ground truth line is matched to the best fitting result line. If one string is shorter than the other, the longer string is split into three parts and the non-matched parts are reprocessed later.

Lines that cannot be matched are treated as either insertion errors or deletion errors.

Different matching strategies are used and the one with the highest success is reported as the flex character accuracy.

In the most trivial cases this measure is equivalent to the traditional character accuracy with the exception that line breaks are disregarded.

Output values:

Name	**Range**	**Description**
flexCharacterAccuracy	[0.0...1.0]	Character accuracy

charsInGroundTruth	>= 0	Number of characters in the ground truth text
charsInResult	>= 0	Number of characters in the result text

Word Accuracy

The word accuracy is the result of the comparison of two lists of words. Two words have to match 100% to be considered equal. Based on the edit distance d the accuracy is defined by:

$$accuracy = \max(0, \frac{m-d}{m})$$

where m is the number of the ground truth words.

The cost function for the edit distance is set to (1,1,1) by default but can be changed to any supported cost function.

A list of stop-words can be provided to filter the lists of input words. Stop-words are words that provide little information and can therefore be ignored. Some English stop-words:

the, to, from, on, for, ...

Output values:

Name	**Range**	**Description**
wordAccuracy	[0.0...1.0]	Word accuracy (including stop-words)
wordsInGroundTruth	>= 0	Number of words in the ground truth text (including stop-words)
wordsInResult	>= 0	Number of words in the result text (including stop-words)
wordAccuracyExclStopWords	[0.0...1.0]	Word accuracy (excluding stop-words)
wordsInGroundTruthExclStopWords	>= 0	Number of words in the ground truth text (excluding stop-words)
wordsInResultExclStopWords	>= 0	Number of words in the result text (excluding stop-words)

Using a Text Replacement Filter

It is possible to apply a filter to the text input files. The text filter contains replacement rules to replace single Unicode characters or a sequence of characters by another character or sequence. To apply a filter use -text-filter in the command line call and provide the file name of the XML file containing the filter rules as additional argument. The XML file must have the following format:

```
<?xml version="1.0" encoding="utf-8"?>
<Parameters>
  <Parameter type="4" name="Replacement Rule"
             id="1"
             sortIndex="1"
             value="0065:=0061"
             visible="false"
             isSet="true">
    <Description>Replace e by a</Description>
  </Parameter>
  <Parameter type="4" name="Replacement Rule"
             id="2"
             sortIndex="2"
             value="0074:="
             visible="false"
             isSet="true">
    <Description>Delete all t</Description>
  </Parameter>
</Parameters>
```

Each parameter element contains a replacement rule. The sortIndex attribute specifies in which order the rules will be applied. The id attribute must be unique (easiest to use the same value as the sort index). The description is optional but helps to understand the rules. The actual rule is encoded in the value attribute. The general format is “HHHH[,HHHH,...]:=[HHHH,HHHH,...]”. HHHH is a Unicode character represented as 4 digit hexadecimal number. In the example above “0065:=0061” means ‘replace all characters e with character a’. To replace a character sequence separate the single characters by comma. The same applies for the right-hand side (the replacement character or sequence). It is also possible to remove characters by leaving the right-hand side empty (e.g. “0074:=” to delete all ts).
For an example how to use a filter, see the example ‘Filter Rules’ in the help folder of the tool. The example includes an XML file with replacement rules and a batch file to process a document.

Special Rules

There is a set of special rules for more complex replacements that cannot be described by the default rule syntax. These rules are applicable by using predefined keywords on the left side of a rule (before the :=). See following table for the keywords and their function:

Keyword for Special Rule	Description
MULTSPACE	Stands for a variable number of consecutive spaces (two or more)
STARTSPACE	Stands for a space at the beginning of a text
ENDSPACE	Stands for a space at the end of a text
MULTBREAK	Stands for a variable number of consecutive line breaks (two or more)

STARTBREAK	Stands for a line break at the beginning of a text
ENDBREAK	Stands for a line break at the end of a text
REGEX####	Generic regular expression (see Java documentation). Note: This rule is not supported by the "Page Converter and Validator" tool for Windows.

Examples:
"_MULTSPACE_:=0020" Replaces all multiple spaces by one space
"_STARTSPACE_:=" Removes a space at the beginning of a text
"_REGEX_\p{Punct}:=" Removes all punctuations

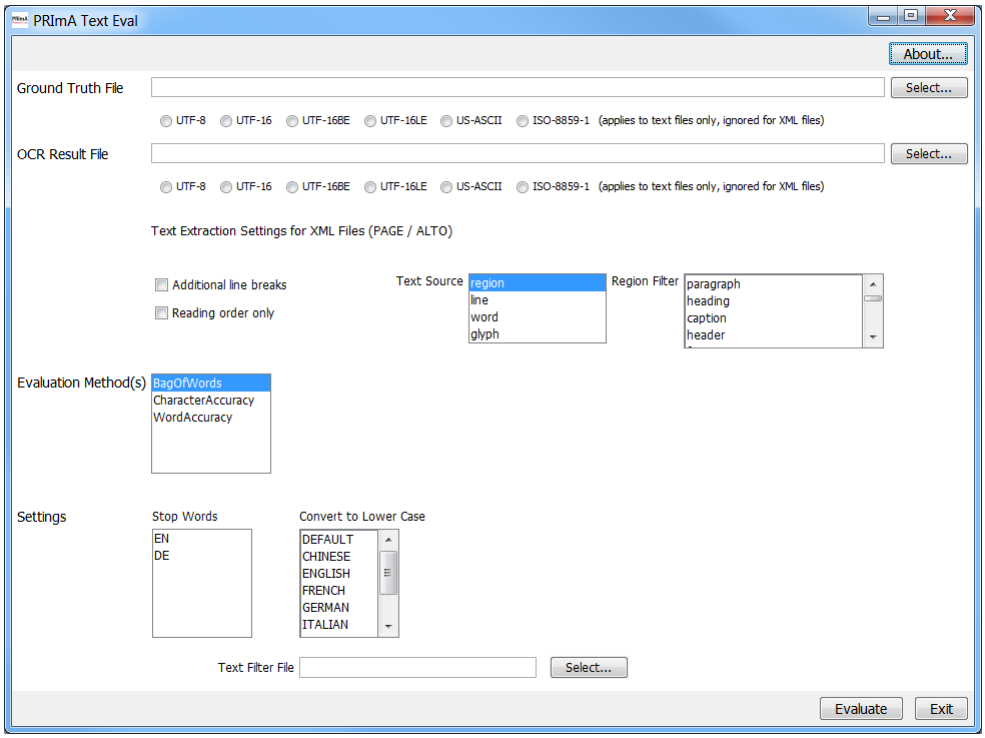
Page Element Filter

Rules can be restricted to individual text element levels (text region, text line, word and glyph) by adding a filter rule at the end of a replacement rule. The extended rule format is defined as: "HHHH[,HHHH,...]:=[HHHH,HHHH,...][|[R][L][W][G]]". Thus, a text element filter can be applied by adding "|" followed by a combination of "R" (region), "L" (text line), "W" (word) and "G" (glyph). The text replacement rule will be carried out only for the specified text element levels.

Example: "0020:=|WG" Removes all spaces from the text of words and glyphs.

Interfaces

The tool is platform independent and comes with a basic graphical user interface (see below) as well as a command line interface (see additional help file).



Credits

PRImA Research Lab
University of Salford
United Kingdom

www.primaresearch.org

Director of research group	*Apostolos Antonacopoulos*
Text evaluation project lead	*Christian Clausner*
Development	*Christian Clausner*
Documentation	*Christian Clausner*