Comparison of Visual and Logical Character Segmentation in Tesseract OCR Language Data for Indic Writing Scripts

Jennifer Biggs
National Security & Intelligence, Surveillance & Reconnaissance Division
Defence Science and Technology Group
Edinburgh, South Australia
{firstname.lastname@dsto.defence.gov.au}

Abstract

Language data for the Tesseract OCR system currently supports recognition of a number of languages written in Indic writing scripts. An initial study is described to create comparable data for Tesseract training and evaluation based on two approaches to character segmentation of Indic scripts; logical vs. visual. Results indicate further investigation of visual based character segmentation language data for Tesseract may be warranted.

1 Introduction

The Tesseract Optical Character Recognition (OCR) engine originally developed by Hewlett-Packard between 1984 and 1994 was one of the top 3 engines in the 1995 UNLV Accuracy test as “HP Labs OCR” (Rice et al 1995). Between 1995 and 2005 there was little activity in Tesseract, until it was open sourced by HP and UNLV. It was re-released to the open source community in August of 2006 by Google (Vincent, 2006), hosted under Google code and GitHub under the *tesseract-ocr* project.[1] More recent evaluations have found Tesseract to perform well in comparisons with other commercial and open source OCR systems (Dhiman and Singh. 2013; Chattopadhyay et al. 2011; Heliński et al. 2012; Patel et al. 2012; Vijayarani and Sakila. 2015). A wide range of external tools, wrappers and add-on projects are also available including Tesseract user interfaces, online services, training and training data preparation, and additional language data.

Originally developed for recognition of English text, Smith (2007), Smith et al (2009) and Smith (2014) provide overviews of the Tesseract system during the process of development and internationalization. Currently, Tesseract v3.02 release, v3.03 candidate release and v3.04 development versions are available, and the *tesseract-ocr* project supports recognition of over 60 languages.

Languages that use Indic scripts are found throughout South Asia, Southeast Asia, and parts of Central and East Asia. Indic scripts descend from the Brāhmī script of ancient India, and are broadly divided into North and South. With some exceptions, South Indic scripts are very rounded, while North Indic scripts are less rounded. North Indic scripts typically incorporate a horizontal bar grouping letters.

This paper describes an initial study investigating alternate approaches to segmenting characters in preparing language data for Indic writing scripts for Tesseract; logical and a visual segmentation. Algorithmic methods for character segmentation in image processing are outside of the scope of this paper.

2 Background

As discussed in relation to several Indian languages by Govandaraju and Stelur (2009), OCR of Indic scripts presents challenges which are different to those of Latin or Oriental scripts. Recently there has been significantly more progress, particularly in Indian languages (Krishnan et al 2014; Govandaraju and Stelur. 2009; Yadav et al. 2013). Sok and Taing (2014) describe recent research in OCR system development for Khmer, Pujari and Majhi (2015) provide a survey

[1] The *tesseract-ocr* project repository was archived in August 2015. The main repository has moved from https://code.google.com/p/tesseract-ocr/ to https://github.com/tesseract-ocr

of Odia character recognition, as do Nishad and Bindu (2013) for Malayalam.

Except in cases such as Krishnan et al. (2014), where OCR systems are trained for whole word recognition in several Indian languages, character segmentation must accommodate inherent characteristics such as non-causal (bidirectional) dependencies when encoded in Unicode.[2]

2.1 Indic scripts and Unicode encoding

Indic scripts are a family of abugida writing systems. Abugida, or alphasyllabary, writing systems are partly syllabic, partly alphabetic writing systems in which consonant-vowel sequences may be combined and written as a unit. Two general characteristics of most Indic scripts that are significant for the purposes of this study are that:

- Diacritics and dependent signs might be added above, below, left, right, around, surrounding or within a base consonant.
- Combination of consonants without intervening vowels in ligatures or noted by special marks, known as consonant clusters.

The typical approach for Unicode encoding of Indic scripts is to encode the consonant followed by any vowels or dependent forms in a specified order. Consonant clusters are typically encoded by using a specific letter between two consonants, which might also then include further vowels or dependent signs. Therefore the visual order of graphemes may differ from the logical order of the character encoding. Exceptions to this are Thai, Lao (Unicode v1.0, 1991) and Tai Viet (Unicode v5.2, 2009), which use visual instead of logical order. New Tai Lue has also been changed to a visual encoding model in Unicode v8.0 (2015, Chapter 16). Complex text rendering may also contextually shape characters or create ligatures. Therefore a Unicode character may not have a visual representation within a glyph, or may differ from its visual representation within another glyph.

2.2 Tesseract

As noted by White (2013), Tesseract has no internal representations for diacritic marks. A typical OCR approach for Tesseract is therefore to train for recognition of the combination of characters including diacritic marks. White (2013) also notes that diacritic marks are often a common source of errors due to their small size and distance from the main character, and that training in a combined approach also greatly expands the larger OCR character set. This in turn may also increase the number of similar symbols, as each set of diacritic marks is applied to each consonant.

As described by Smith (2014), lexical resources are utilised by Tesseract during two-pass classification, and de Does and Depuydt (2012) found that word recall was improved for a Dutch historical recognition task by simply substituting the default Dutch Tesseract v3.01 word list for a corpus specific word list. As noted by White (2013), while language data was available from the *tesseract-ocr* project, the associated training files were previously available. However, the Tesseract project now hosts related files from which training data may be created.

Tesseract is flexible and supports a large number of control parameters, which may be specified via a configuration file, by the command line interface, or within a language data file[3]. Although documentation of control parameters by the *tesseract-ocr* project is limited[4], a full list of parameters for v3.02 is available[5]. White (2012) and Ibrahim (2014) describe effects of a limited number of control parameters.

2.2.1 Tesseract and Indic scripts

Training Tesseract has been described for a number of languages and purposes (White, 2013; Mishra et al. 2012; Ibrahim, 2014; Heliński et al. 2012). At the time of writing, we are aware of a number of publically available sources for Tesseract language data supporting Indic scripts in addition to the *tesseract-ocr* project. These include *Parichit*[6], *BanglaOCR*[7] (Hasnat et al. 2009a and 2009b; Omee et al. 2011) with training files released in 2013, *tesseractindic*[8], and *myaocr*[9]. Their Tesseract version and recognition languages are summarised in Table 1. These external projects also provide Tesseract training data in the form of TIFF image and associated coordinate 'box' files. For version 3.04, the *tesseract-ocr* project provides data from which Tesseract can generate training data.

[2] Except in Thai, Lao, Tai Viet, and New Tai Lue

[3] Language data files are in the form <xxx>.traineddata

[4] https://code.google.com/p/tesseract-ocr/wiki/ControlParams

[5] http://www.sk-spell.sk.cx/tesseract-ocr-parameters-in-302-version

[6] https://code.google.com/p/Parichit/

[7] https://code.google.com/p/banglaocr/

[8] https://code.google.com/p/tesseractindic/

[9] https://code.google.com/p/myaocr/

Sets of Tesseract language data for a given language may differ significantly in parameters including coverage of the writing script, fonts, number of training examples, or dictionary data.

Project	v.	Languages
tesseract-ocr	3.04	Assamese, Bengali, Gujarati, Hindi, Marathi, Odia, Punjabi, Tamil, Myanmar, Khmer, Lao, Thai, Sinhala, Malayalam, Kannada, Telugu
	3.02	Bengali, Tamil, Thai
	3.01	Hindi, Thai
myaocr	3.02	Myanmar
Parichit	3.01	Bengali, Gujarati, Hindi, Oriya, Punjabi, Tamil, Malayalam, Kannada, Telugu
tesseractindic	2.04	Hindi, Bengali, Malayalam
BanglaOCR	2	Bengali

Table 1: Available Indic language data for Tesseract

Smith (2014)[10] and Smith et al (2009)[11] provides results for Tesseract for two Indic scripts; Hindi[12] and Thai. Table 2 compares these error rates to those found by Krishnan et al. (2014)[13]. Additionally, the Khmer OCR project reports initial accuracy rates of 50-60% for Khmer OS Battambang font, 26pt (Tan, 2014), and the Khmer OCR project[14] beta website provides a Khmer OCR web service based on the Tesseract OCR system that incorporates user feedback training. Hasnat et al. (2009a; 2009b) report on development of Bengali language data for BanglaOCR, with 70-93% accuracy depending on image type. Omee et al. (2011) report up to 98% accuracy in limited contexts for BanglaOCR. Nayak and Nayak (2014) report on development of Odia language data with 98-100% recognition accuracy for isolated characters.

Language	Ground truth (million)		Error rate (%)	
	char	words	char	word
Hindi *	-	0.39	26.67	42.53
Telugu *	-	0.2	32.95	72.11
Hindi **	2.1	0.41	6.43	28.62
Thai **	0.19	0.01	21.31	80.53
Hindi ***	1.4	0.33	15.41	69.44

Table 2: Tesseract error rates * from Krishnan et al. (2014) ** from Smith (2014) * from Smith et al (2009)**

2.2.2 Visual and logical character segmentation for Tesseract

As noted by White (2013) the approach of the *tesseract-ocr* project is to train Tesseract for recognition of combinations of characters including diacritics. For languages with Indic writing scripts, this approach may also include consonant-vowel combinations and consonant clusters with other dependent signs, and relies on character segmentation to occur in line with Unicode logical ordering segmentation points for a given segment of text. An advantage of this approach is that Unicode standard encoding is output by the OCR system.

An alternate approach in developing a training set for Tesseract is to determine visual segmentation points within the writing script. This approach has been described and implemented in several external language data projects for Tesseract, including *Parichit*, *BanglaOCR*, and *myaocr*. Examples of logical and two possible approaches to visual segmentation for selected consonant groupings are shown in Figure 1. A disadvantage of visual segmentation is that OCR text outputs may require re-ordering processing to output Unicode encoded text.

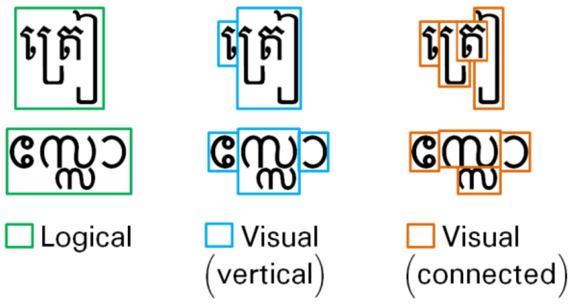

Figure 1: Comparison of logical and two possible visual segmentation approaches for selected characters

[10] Tesseract v3.03 or v3.04
[11] Tesseract v3.00
[12] Hindi and Arabic language data for Tesseract v3.02 used a standard conventional neural network character classifier in a 'cube' model. Although, Smith (2014) states that this model achieves ~50% reduction in errors on Hindi when run together with Tesseract's word recognizer, the training code is unmaintained and unutilised, and will be removed from future *tesseract-ocr* versions.
[13] Tesseract v3.02
[14] The Khmer OCR project led by Mr. Danh Hong begun in 2012 is described by Mr. Ly Sovannra in Tan (2014) and at http://www.khmertype.org

Mishra et al. (2012) describe creating language data for Hindi written in Devanagari script that implemented a visual segmentation approach in which single touching conjunct characters are excluded from the training set. Therefore, Tesseract language data could be created that included only two or more touching conjunct characters, basic characters and isolated half characters. This had the effect of reducing the Tesseract training set[15] and language data size, and increasing recognition accuracy on a test set of 94 characters compared with the *tesseract-ocr* (Google) and *Parichit* language data as shown in Table 3.[16]

Language data	**Training set size**	**Accuracy (%)**
tesseract-ocr v3.01	1729	45.2
Parichit	2173	22.3
Mishra et al. (2012)	786	90.9

Table 3: Comparison of training set, language data and accuracy from Mishra et al. (2012)

The implementation also included language-specific image pre-processing to 'chop' the *Shirorekha* horizontal bar connecting characters within words. This was intended to increase the likelihood of Tesseract system segmentation occurring at these points. Examples of words including *Shirorekha* are shown in Figure 2.

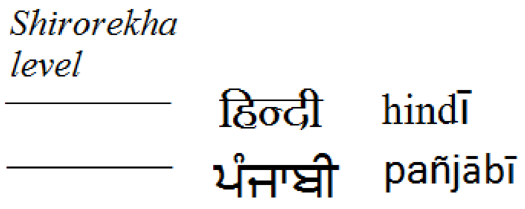

Figure 2: Examples of *Shirorekha* in Devanagari and Gurmukhi scripts

3 Comparison of visual and logical segmentation for Tesseract

An initial study was conducted to determine the potential of implementing a visual segmentation approach, compared to the logical segmentation approach in Tesseract for languages with Indic scripts. Languages written with Indic scripts that do not use the *Shirorekha* horizontal bar were considered. Re-ordering of OCR text outputs for visual segmentation methods is outside the scope of this study. The term glyph is used in this section to describe a symbol that represents an OCR recognition character, whether by logical or visual segmentation.

3.1 Method

This section describes ground truth and evaluation tools used, and the collection and preparation of glyph, Tesseract training, and OCR ground truth data. Three Indic languages were selected to estimate the potential for applying visual segmentation to further languages. Firstly, corpora were collected and analysed to compare glyphs found by each segmentation approach. Secondly, Tesseract recognition and layout accuracy was evaluated based on the coverage of those glyphs in the corpus. The accuracy of *tesseract-ocr* project v3.04 language data is also measured against the same ground truth data for a wider selection of Indic languages.

3.1.1 Glyph data

In order to estimate the number and distribution of glyphs in selected Indic languages, language specific corpora were sought. A web crawler was implemented using the *crawler4j* library[17], which restricted the crawl domain to the seed URL. The *boilerpipe* library [18] was then used to extract textual content from each web page. For each language, a corpus was then collected by using the relevant Wikipedia local language top page as the seed for the crawler.

The *Lucene* library[19] was used to index corpus documents. Language specific processing was implemented supporting grouping of consonant-vowel combinations, consonant clusters and dependent signs into logical order glyphs. Additional processing to separate those groupings in line with the visual segmentation approach was also implemented.

Letters affected by visual segmentation in each language are shown in Table 4. In Khmer, there could theoretically be up to three *coeng* (U+17D2) in a syllable; two before and one after a vowel. Clusters with *coeng* after a vowel were not additionally segmented in this implementation. The number of glyphs according to each segmentation approach was then extracted from the index for each language. Similarly, in Mala-

[15] Defined in Tesseract the *.unicharset file within language data

[16] It is not stated if text output re-ordering processing for *Parichit* recognition output was applied before accuracy was measured.

[17] https://github.com/yasserg/crawler4j

[18] https://github.com/kohlschutter/boilerpipe

[19] https://lucene.apache.org/core/

yalam dependent vowels found between consonants in consonant ligatures were not segmented.

Language	Letters
Khmer	◌ើ ◌ឿ ◌ៀ ◌េ ◌ែ ◌ៃ ◌ះ ◌ៈ [U+17BE - U+17C3, U+17C7, U+17C8] ◌ោ ◌ៅ (left components) [U+17C4 and U+17C5]
Malayalam	◌ം ◌ഃ ◌ാ ◌ി ◌ീ ◌ു ◌ൂ ◌ൃ ◌ൄ ◌െ ◌േ ◌ൈ ◌ൊ ◌ോ ◌ൌ ◌ൗ [U+0D02, U+0D03, U+0D3E - U+0D4C, U+0D57]
Odia	◌ଂ ◌ଃ ◌ା ◌ୀ ◌େ ◌ୈ ◌ୋ ◌ୌ [U+0B02, U+0B03, U+0B3E, U+0B40, U+0B47 - U+0B4C]

Table 4: Letters and consonant clusters affected by visual segmentation processing per language

The size of corpus and number of glyphs according to logical segmentation is given in Table 5.

Language	Text corpus (Mb)	Logical glyphs (million)
Khmer	252	137.0
Malayalam	307	134.8
Odia	68.9	96.6

Table 5: Text corpus size and occurrences of logical glyphs per language

3.1.2 Tesseract training data

Tesseract training data was prepared for each language using the paired sets of glyph data described in section 3.1. An application was implemented to automatically create Tesseract training data from each glyph data set, with the ability to automatically delete dotted consonant outlines displayed when a Unicode dependent letter or sign is rendered separately. The implemented application outputs multi-page TIFF format images and corresponding bounding box coordinates in the Tesseract training data format.[20]

Tesseract training was completed using most recent release v3.02 according to the documented training process for Tesseract v3, excluding shapeclustering. The number of examples of each glyph, between 5 and 40 in each training set, was determined by relative frequency in the corpus. A limited set of punctuation and symbols were also added to each set of glyph data, equal to those included in *tesseract-ocr* project language data. However, training text was not representative as recommended in documentation, with glyphs and punctuation randomly sorted.

3.1.3 Dictionary data

As dictionary data is utilised during Tesseract segmentation processing, word lists were prepared for each segmentation approach. As the separated character approach introduced a visual ordering to some consonant-vowel combinations and consonant clusters, word lists to be used in this approach were re-ordered, in line with the segmentation processing used for each language described in section 3.1. Word lists were extracted from the *tesseract-ocr* project v3.04 language data.

3.1.4 Ground truth data

OCR ground truth data was prepared in a single font size for each language in the PAGE XML format (Pletschacher and Antonacopoulos. 2010) using the application also described in section 3.1.2. The implementation segments text according to logical or visual ordering described in section 3.1.1, and uses the *Java PAGE libraries*[21] to output PAGE XML documents.

Text was randomly selected from documents within the web corpora described in section 3.1. Text segments written in Latin script were removed. Paired ground truth data were then generated. For each document image, two corresponding ground truth PAGE XML files were created according to logical and visual segmentation methods.

3.1.5 Evaluation

Tesseract v3.04 was used via the *Aletheia* v3 tool for production of PAGE XML ground truth described by Clausner et al. (2014). Evaluation was completed using the layout evaluation framework for evaluating PAGE XML format OCR outputs and ground truth described by Clausner et al. (2011). Output evaluations were completed using the described *Layout Evaluation* tool and stored in XML format.

[20] Description of the training format and requirements can be found at https://github.com/tesseract-ocr/tesseract/wiki/TrainingTesseract

[21] The PAGE XML format and related tools have been developed by the PRImA Research Lab at the University of Salford, and are available from http://www.primaresearch.org/tools/

3.2 Results

Results are presented in three sections; for *tesseract-ocr* language data, for web corpora glyph data per segmentation method, and for the comparable Tesseract language data per segmentation method.

Measured layout success is a region correspondence determination. Results are given for glyph based count and area weighted arithmetic and harmonic mean layout success as calculated by the *Layout Evaluation* tool. Weighted area measures are based on the assumption that bigger areas regions are more important than smaller ones, while the weighted count only takes into account the error quantity.

3.2.1 *Tesseract-ocr* language data

Recognition accuracy for selected *tesseract-ocr* project language data with Indic scripts is given in Table 6. All glyphs are segmented in line with Unicode logical encoding standards; using a logical segmentation approach, except for Thai and Lao which are encoded with visual segmentation in Unicode.

Measured Thai recognition accuracy is in line with the 79.7% accuracy reported by Smith (2014). While Hindi accuracy is far less than the 93.6% reported by Smith (2014), it is higher than the 73.3% found by Krishnan et al. (2014). Measured recognition accuracy for Telugu is also higher than the 67.1% found by Krishnan et al. (2014), although this may be expected for higher quality evaluation images. Measured Khmer recognition accuracy is in line with the 50-60% reported in Tan (2014). Bengali results are within the 70-93% range reported by Hasnat et al. (2009a), but are not directly comparable with the training approach used in *BanglaOCR*.

3.2.2 Web corpora glyphs by logical and visual segmentation

The number of glyphs and their occurrences in the collected language specific Wikipedia corpora are shown in Figure 4. These are compared to the number of glyphs in the *tesseract-ocr* project language data recognition character set[22], and the number of glyphs when visual order segmentation processing is applied to that character set. Visual segmentation can be seen to significantly reduce the number of glyphs for the same language coverage in each case. The logical glyphs in common and unique to *tesseract-ocr* and corpus based language data may be seen in Figure 3.

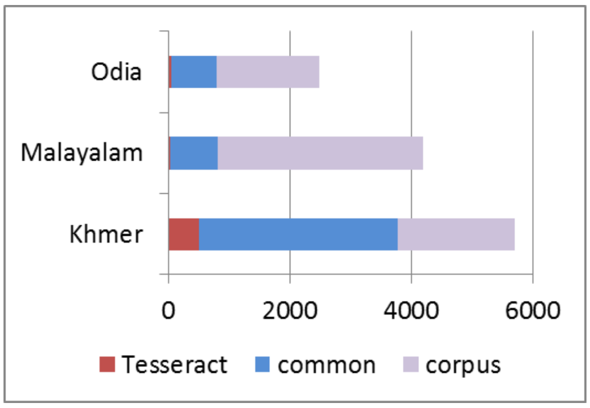

Figure 3: Coverage of logical glyphs between *tesseract-ocr* and corpus based language data

3.2.3 Comparable data for logical and visual segmentation

The total number of examples in the training data and size of the resulting Tesseract language data file with each approach (without dictionary data) is given in Table 7. The *tesseract-ocr* language data sizes are not directly comparable as the training sets and fonts differ.

OCR recognition accuracy is given for each segmentation method in Table 7. Recognition accuracy was found to be higher for visual segmentation in each language; by 3.5% for Khmer, 16.1% for Malayalam, and by 4.6% for Odia.

Logical segmentation accuracy shown in Table 7 was measured against the same ground truth data reported in section 3.2.1. However, as illustrated in Figure 4, the coverage of glyphs in each set of language data differed greatly. In each case, the number of glyphs found in the collected corpus was significantly greater than in the *tesseract-ocr* recognition set.

Recognition accuracy for *tesseract-ocr* language data for Khmer and Malayalam was 12.2% and 13% higher respectively than for the corpus based logical segmentation language data when measured against the same ground truth. However the corpus based logical segmentation data for Odia achieved 12.2% higher recognition accuracy than *tesseract-ocr* language data.

Dictionary data added to language data for each segmentation method was found to make no more than 0.5% difference to recognition or layout accuracy for either segmentation method.

[22] Glyphs not within the local language Unicode range(s) are not included.

Language	Recognition accuracy (%)	Mean overall layout success (%) Area weighted Arith.	Har.	Count weighted Arith.	Har.	Ground truth Glyphs (logical)	Char	Recognition glyphs
Assamese	26.1	65.3	49.6	59.5	47.2	1080	1795	1506
Bengali	71.8	92.7	91.9	66.8	63.5	1064	1932	1451
Khmer	52.2	92.6	92.1	82.9	81.0	556	1099	3865
Lao *	77.1	96.6	96.5	85.6	84.1	1139	1445	1586
Gujarati	1.8	69.6	64.2	57.6	53.1	974	1729	1073
Hindi	81.9	89.1	87.4	58.2	49.4	952	1703	1729
Malayalam	62.7	90.6	89.2	82.5	78.1	552	1153	855
Myanmar	25.6	86.8	84.4	67.2	59.2	598	1251	7625
Odia	63.7	96.3	96.1	90.0	88.7	864	1514	834
Punjabi **	0.1	61.4	41.6	65.4	52.3	916	1569	1029
Tamil	89.2	95.5	95.0	93.1	92.4	798	1290	295
Telugu	75.3	78.0	72.6	55.1	44.2	877	1674	2845
Thai *	79.7	95.1	94.7	86.7	85.7	1416	1727	864

Table 6: Glyph recognition and layout accuracy for *tesseract-ocr* project v3.04 language data for selected Indic languages *languages encoded in visual segmentation in Unicode ** written in Gurmukhi script

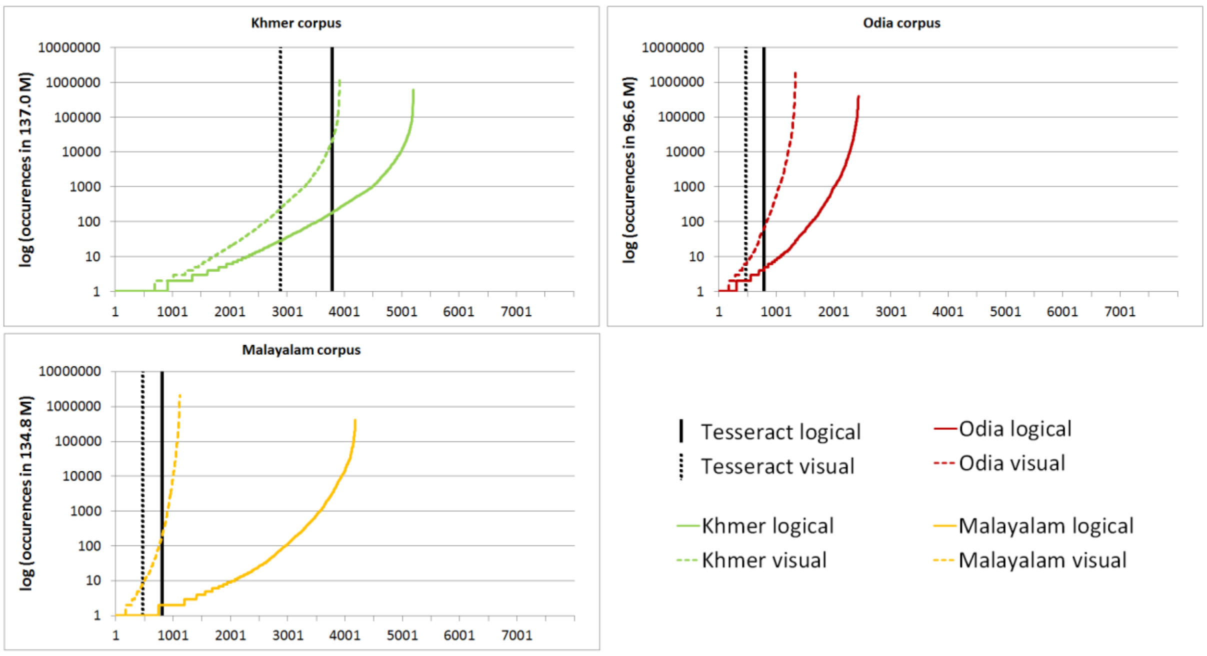

Figure 4: Comparison of logical vs. visual segmentation of glyphs in corpora

Language	Seg-menta-tion	Recogni-tion accu-racy (%)	Mean overall layout success (%) Area weighted Arith.	Har.	Count weighted Arith.	Har.	Ground truth glyphs	Recognition glyphs
Khmer	Logical	41.0	92.8	91.9	83.6	80.5	556	5205
	Visual	44.5	92.9	92.3	86.9	85.8	677	3965
Malayalam	Logical	54.2	90.2	88.4	80.4	74.3	552	4237
	Visual	70.3	90.8	89.7	80.5	77.6	851	1171
Odia	Logical	75.9	94.8	94.4	88.2	86.4	864	2491
	Visual	80.5	95.1	94.7	91.5	90.8	1130	1387

Table 7: Glyph recognition and layout accuracy, ground truth and language data for logical and visual segmentation

4 Discussion

Analysis of the collected glyph corpora and *tesseract-ocr* project language data has shown the visual segmentation significantly reduces the number of glyphs required for a Tesseract training set in each of the languages considered. When using comparative training and ground truth data, visual segmentation was also shown to reduce the size of Tesseract language data and increase recognition accuracy. The use of dictionary data was not found to significantly affect results.

The implementation for visual segmentation of glyphs led to inconsistencies between similar visual components. For example, in Khmer it was observed that the visual representation of *coeng* (U+17D2) was commonly segmented by Tesseract as a separate glyph using *tesseract-ocr* and created language data, as illustrated for Khmer in Figure 5. Further opportunities for visual segmentation were also not implemented, such as components of consonant clusters. A consistent and more sophisticated implementation of visual segmentation may further improve results.

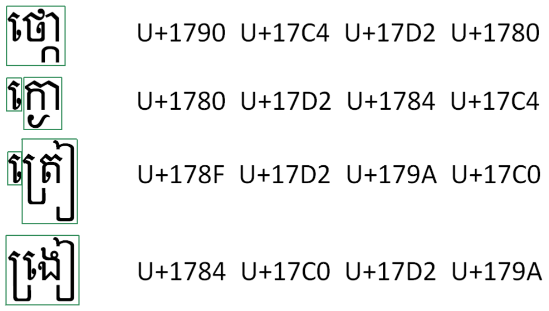
U+1790 U+17C4 U+17D2 U+1780
U+1780 U+17D2 U+1784 U+17C4
U+178F U+17D2 U+179A U+17C0
U+1784 U+17C0 U+17D2 U+179A

Figure 5: Visual glyphs for Khmer as implemented

The Tesseract training data prepared from corpus based glyphs was intended to be comparable, but was not in line with recommendations for training Tesseract. Preparation of training data in line with recommendations may improve results. The effects of Tesseract configuration parameters were not investigated during this study and should also be explored per language. Further, while glyph recognition accuracy achieved for the visual segmentation language data for Khmer was lower than that of the *tesseract-ocr* project language data, the coverage of glyphs was far greater. A significant percentage of the glyphs in each training set were rare. Future work may examine the relationship between coverage of rare glyphs in language data and recognition accuracy.

While effort was made to estimate coverage of modern glyphs for each segmentation approach in each language, the web corpora collected may not be representative. In preparing training data for the proposed segmentation method, care must be taken to determine that isolated or combined characters in the training sets are rendered in the predicted way when combined with other characters. A further consideration when creating multi-font training data is that characters may be rendered significantly differently between fonts. Further, some scripts have changed over time. For example, Malayalam has undergone formal revision in the 1970s, and informal changes with computer-aided typesetting in the 1980s, and Devanagari has also modified specific characters during the last three decades.

5 Conclusion

Developing high accuracy, multi-font language data for robust, end-to-end processing for Tesseract was not within the scope of this study. Rather, the aim was an initial investigation of alternate approaches for logical compared to visual character segmentation in a selection of Indic writing scripts. Results in the limited evaluation domain indicate that the proposed visual segmentation method improved results in three languages. The described technique may potentially be applied to further Indic writing scripts. While recognition accuracy achieved for the reported languages remains relatively low, outcomes indicate that effort to implement language specific training data preparation and OCR output reordering may be warranted.

Acknowledgements

We would like to acknowledge Google and the work of Mr. Ray Smith and contributors to the *tesseract-ocr* project.

We would also like to acknowledge the PRImA Lab of the University of Salford for their work in developing the PAGE XML format and related software tools and applications.

References

Chattopadhyay, T., Sinha, P., and Biswas, P. *Performance of document image OCR systems for recognizing video texts on embedded platform*. International Conference on Computational Intelligence and Communication Networks (CICN), 2011, pp. 606-610, Oct 2011

Clausner, C., Pletschacher, S. and Antonacopoulos, A. 2014. *Scenario Driven In-Depth Performance Evaluation of Document Layout Analysis Methods*. In proc. of the 11th International Conference on Document Analysis and Recognition (ICDAR2011), Beijing, China, September 2011, pp. 1404-1408

Clausner, C., Pletschacher, S. and Antonacopoulos, A. 2014. *Efficient OCR Training Data Generation with Aletheia*. Short paper booklet of the 11th International Association for Pattern Recognition Workshop on Document Analysis Systems (DAS2014), Tours, France, April 2014, pp. 19-20

de Does, Jesse. and Depuydt, Katrien. 2012. *Lexicon-supported OCR of eighteenth century Dutch books: a case study*. In proc. SPIE 8658, Document Recognition and Retrieval XX, 86580L (February 4, 2013); doi:10.1117/12.2008423

Dhiman and Singh. 2013. *Tesseract Vs Gocr A Comparative Study*. International Journal of Recent Technology and Engineering (IJRTE): Vol 2, Issue 4, September 2013

Govindaraju, Venugopal, and Srirangaraj Setlur. 2009. *Guide to OCR for Indic scripts: document recognition and retrieval.* London: Springer.

Hasnat, Abul., Chowdhury, Muttakinur Rahman. and Khan, Mumit. 2009a. *An open source Tesseract based Optical Character Recognizer for Bangla script*. In proc. Tenth International Conference on Document Analysis and Recognition (ICDAR2009), Catalina, Spain, July 26-29, 2009

Hasnat, Abul., Chowdhury, Muttakinur Rahman. and Khan, Mumit. 2009b. *Integrating Bangla script recognition support in Tesseract OCR*. In proc. Conference on Language Technology 2009 (CLT09), Lahore, Pakistan, January 22-24, 2009

Heliński, Marcin., Kmieciak, Miłosz. and Parkoła, Tomasz. 2012. *Report on the comparison of Tesseract and ABBYY FineReader OCR engines*. IMPACT Report. http://www.digitisation.eu/download/IMPACT_D-EXT2_Pilot_report_PSNC.pdf Last Accessed 3/9/2015

Ibrahim, Ahmed. 2014. *Dhivehi OCR: Character Recognition of Thaana Script using Machine Generated Text and Tesseract OCR Engine*, Edith Cowan University, Australia. http://www.villacollege.edu.mv/iri/images/thaana2.pdf, Last accessed 3/9/2015

Krishnan, Praveen., Sankaran, Naveen., Singh, and Ajeet Kumar. 2014. *Towards a Robust OCR System for Indic Scripts*, 11th IAPR International Workshop on Document Analysis Systems (DAS2014), Tours, France, 7th – 10th April 2014.

Mishra, Nitin; Patvardhan, C.; Lakshimi, Vasantha C.; and Singh, Sarika. 2012. *Shirorekha Chopping Integrated Tesseract OCR Engine for Enhanced Hindi Language Recognition*, International Journal of Computer Applications, Vol. 39, No. 6, February 2012, pp. 19-23

Nishad, A; and Bindu, K. 2013. *Malayalam OCR Systems – A Survey*. International Journal of Computer Technology and Electronics Engineering, Vol. 3, No. 6, December 2013

Nayak, Mamata. and Nayak, Ajit Kumar. 2014. *Odia Characters Recognition by Training Tesseract OCR Engine*. In proc. International Conference in Distributed Computing and Internet Technology 2014 (ICDCIT-2014)

Omee, Farjana Yeasmin., Himel, Shiam Shabbir. and Bikas, Md. Abu Naser. 2011. *A Complete Workflow for Development of Bangla OCR*. International Journal of Computer Applications, Vol. 21, No. 9, May 2011

Patel, Chirag., Patel, Atul and Patel, Dharmendra. 2012. *Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study*, International Journal of Computer Applications, Vol. 55, No. 10

Pletschacher, S. and Antonacopoulos, A. 2010. *The PAGE (Page Analysis and Ground-Truth Elements) Format Framework*. In proc. of the 20th International Conference on Pattern Recognition (ICPR2010), Istanbul, Turkey, August 23-26, 2010, IEEE-CS Press, pp. 257-260

Pujari, Pushpalata; and Majhi, Babita. 2015. *A Survey on Odia Character Recognition*. International Journal of Emerging Science and Engineering (IJESE), Vol. 3, No. 4, February 2015

Rice, S.V., Jenkins, F.R. and Nartker, T.A. 1995. *The Fourth Annual Test of OCR Accuracy*, Technical Report 95-03, Information Science Research Institute, University of Nevada, Las Vegas

Smith, Ray D. Antonova and D. Lee. 2009 *Adapting the Tesseract Open Source OCR Engine for Multilingual OCR*, in proc. International Workshop on Multilingual OCR 2009, Barcelona, Spain, July 25, 2009

Smith, Ray. 2007. *An overview of the Tesseract OCR Engine*, in proc. Of the 9thInternational Conference on Document Analysis and Recognition (ICDAR2007), Curitiba, Paraná, Brazil, 2007

Smith, Ray. 2014. *Everything you always wanted to know about Tesseract.* 11th IAPR International Workshop on Document Analysis Systems (DAS2014), Tours, France, 7th – 10th April 2014. Tutorial slides available from https://drive.google.com/file/d/0B7l10Bj_LprhbUlIUFlCdGtDYkE/view?pli=1 Last visited 11/9/2015

Sok, Pongsametry. and Taing, Nguonly. 2014. *Support Vector Machine (SVM) Based Classifier For Khmer Printed Character-set Recognition*, Asia Pacific Signal and Information Processing Association, 2014, Annual Summit and Conference (APSIPA), 9-12 December, 2014, Siem Reap, city of Ankor Wat, Cambodia

Tan, Germaine. 2014. Khmer OCR: Convert Hard-Copy Khmer Text To Digital. Geeks in Cambodia, November 18, 2014. http://geeksincambodia.com/khmer-ocr-convert-hard-copy-khmer-text-to-digital/ Last visited 13/9/2015

Vijayarani and Sakila. 2015. *Performance Comparison of OCR Tools*. International Journal of UbiComp (IJU), Vol. 6, No. 3, July 2015

Vincent, Luc. 2006. *Announcing Tesseract OCR*, Google Code, http://googlecode.blogspot.com.au/2006/08/announcing-tesseract-ocr.html Last accessed 1/9/2015

White, Nick. 2013. *Training Tesseract for Ancient Greek OCR*. The Eutypon, No. 28-29, October 2013, pp. 1-11. http://ancientgreekocr.org/e29-a01.pdf Last visited 18/9/2015

Yadav, Divakar; Sanchez-Cuadrado, Sonia; and Morato, Jorge. 2013. *Optical Character Recognition for Hindi Language Using a Neural Network Approach*. Journal of Information Processing Systems, Vol. 9, No. 10, March 2013, pp. 117-140