Dhivehi OCR:
Character Recognition of Thaana Script using Machine-Generated Text and Tesseract OCR Engine

Ahmed Ibrahim

Abstract

This paper provides technical aspects and the context of recognising Dhivehi characters using Tesseract OCR Engine, which is a freely available OCR engine with remarkable accuracy and support for multiple languages. The experiments that were conducted showed promising results with 69.46% accuracy and, more importantly, highlighted limitations that are unique to Dhivehi. These issues have been discussed in detail and possible directions for future research are presented.

Keywords: Dhivehi OCR, Thaana Script, Optical Character Recognition, Tesseract OCR.

1. Introduction

Optical Character Recognition (OCR) is the process of mimicking humans' ability to read text. There has been a considerable accumulation of knowledge in this area of computer science. It is still in development, albeit being a mature problem of pattern recognition and artificial intelligence (Mori, Nishida, & Yamada, 1999). There is currently no published literature related to OCR of Dhivehi text, nor are there any tools in the public domain specifically developed to perform Dhivehi OCR.

With the proliferation of Internet and mobile connectivity, the demand for digital content continues to grow. There has been a tremendous increase of Internet content in the Dhivehi language over the recent years. The social media phenomenon has been an influential force behind this drive to publish more localised content for the Maldivian audience. However, developments in Dhivehi language technologies have been very slow and often lacking.

The absence of such literature and tools for Dhivehi OCR has been the primary motivation for this paper. The objective of this paper is to inform the reader of how a freely available tool, Tesseract OCR Engine, can be used for Dhivehi OCR. As a result, key problems intrinsic to the Dhivehi language were identified so that future research can be conducted to improve this domain.

This paper will first present some background information on OCR and basic building blocks of a typical system. It will also look at certain characteristics of Dhivehi text that are relevant to the problem, followed by a brief overview of the Tesseract OCR Engine. In the next section, the methodology used to conduct the experiment has been explained. The section on findings will present the results and the discussion section delves into the successes and failures of the experiment in more detail. The paper is then concluded with a summary and future directions for research.

2. Background

2.1 Optical Character Recognition

Today there are a number of ways of entering data into a computer, in contrast to the traditional form of using the keyboard. In order to increase productivity and accuracy, automatic forms of data capture or Automatic Identification are used more frequently in a number of applications. Some examples of such *Automatic Identification* techniques are scanning barcodes from consumer products and reading magnetic strips from credit cards. These have become trivial because they are seamlessly integrated into our everyday routines. By reducing human intervention, human errors are minimised and efficiency is increased.

Optical Character Recognition, commonly abbreviated as OCR, is a form of Automatic Identification where a computer system converts images of typewritten text into a standard encoding scheme such as ASCII or Unicode, so that it is editable on the computer. OCR is performed off-line after the text has been written or printed as opposed to on-line recognition where the computer recognises the characters as they are drawn. Since its accuracy depends directly on the quality of the input document/image, the more it is constrained, the better it performs. Thus, current OCR technologies are still limited when it comes to unconstrained handwriting (Eikvil, 1993).

According to Mori, Suen, and Yamamoto (1992), the first concepts of OCR date back to even before the age of computers; a patent was filed in Germany by Tauschek (Tauschek, 1935) in 1929 and one was later filed in the U.S. by Handel (Handel, 1933) in 1933. However, the first generation of commercial OCR systems only started appearing in the period 1960 to 1965 (Eikvil, 1993). Since then, OCR has gone through major developments and significant improvements; however, it is far from being "a solved problem." Different fonts and varying qualities of input documents still introduce a wide array of challenges for OCR (Kae, Smith, & Learned-Miller, 2011).

2.2 Components of an OCR System

A typical multilingual OCR system is similar to a pipeline of sequential stages, one after the other, each feeding its output to the next. It most commonly involves the *pre-processing* of the input image, *feature extraction* and c*lassification/recognition* (Peng, Cao, Setlur, Govindaraju, & Natarajan, 2013).

The purpose of pre-processing is to clean, prepare and format the image data so that the features required for the classifier can be accurately extracted. As such, pre-processing may include binarization, layout analysis, line segmentation and word/character segmentation. Binarization eases the recognition process (Elagouni, Garcia, Mamalet, & Sébillot, 2013) by converting the image from either a grayscale or colour profile to a bi-level black and white image. This immensely reduces the amount of data that needs to be processed. The layout analysis typically identifies text and non-text regions in a page and also splits multi-column text into columns (Smith, 2009). This text is then further segmented into individual lines and often into words and/or characters. There is ongoing research in this area to overcome issues such as character fragmentation, noise separation, and distinguishing graphics or geometry from text (Eikvil, 1993).

The feature extraction greatly depends on the type of classifier used for the identification. Hence, features may be represented based on the structural information of the characters,

statistical input patterns of feature vectors, or even a hybrid approach. These features would then be used for structural matching, probabilistic matching, or statistical classification (Cheng-Lin, Jaeger, & Nakagawa, 2004).

2.3 Characteristics of Dhivehi Characters

Dhivehi is the native language spoken in the Maldives and is also the official language used by all government institutions. Its base vocabulary is *Indo-Aryan*; however, many of its words have been borrowed or derived from other languages such as Arabic, English, Dravidian and Sinhala (Gair, 2010). The geographically dispersed nature of speakers of Dhivehi throughout the Maldives has resulted in remarkable dialectal differences. The most notable of those are the *Huvadhoo, Fuammulaku, and Addu* dialects, which are the southernmost atolls in Maldives. However, the *Male*' dialect has become more widespread and is typically spoken predominantly to the north of Hadhdhummathi. Hence, the Male' dialect is considered as the standard and is commonly used in education and administration all over the country (Fritz, 2002).

Despite the different dialects, there is only one standard script used presently, called *Thaana*. According to Fritz (2002), the earliest form of Dhivehi text, *Eveyla Akuru*, can be dated back to 12th century statues found in the National Museum in Male'. More historical artefacts dating back to the 17th century reveal another form of Dhivehi text called *Dives Akuru*. Documents found in later periods revealed the use of Thaana, which was then used in conjunction with Dives Akuru. By the end of the 19th century, Thaana had superseded its predecessors and is now considered the official script of Dhivehi. The formulation of the Thaana script has had significant influence from Arabic text. In fact, the first nine characters of Thaana have a phenomenal resemblance with the numerals used in Arabic (Gair, 2010).
The main characteristics of Thaana writing are:

1. The thaana script consists of 24 basic consonants (Figure 1), 14 Arabic extensions (Figure2) and 11 vowels (Figure 3).
2. It is written from right-to-left except for numbers, which are written from left-to-right.
3. It does not have a distinction between upper and lower case characters.
4. A character sound maybe produced by placing a vowel or diacritic mark above or below a consonant (Figure 4).
5. A consonant maybe written by itself whereas a vowel can never be written by itself.
6. Consequent to the numerous words borrowed from Arabic, any Arabic word can be accurately written semantically using the Arabic extensions found in the Thaana character set. However, they are often interchanged with regular consonants with similar sounds.
7. Due to the shape of characters and different font designs, machine-printed Thaana text can have horizontal overlaps among consonants and also vertical overlaps among consonant-vowel pairs.

8. There is also a great deal of similarity between consonants, between vowels, and also between consonants and vowels.

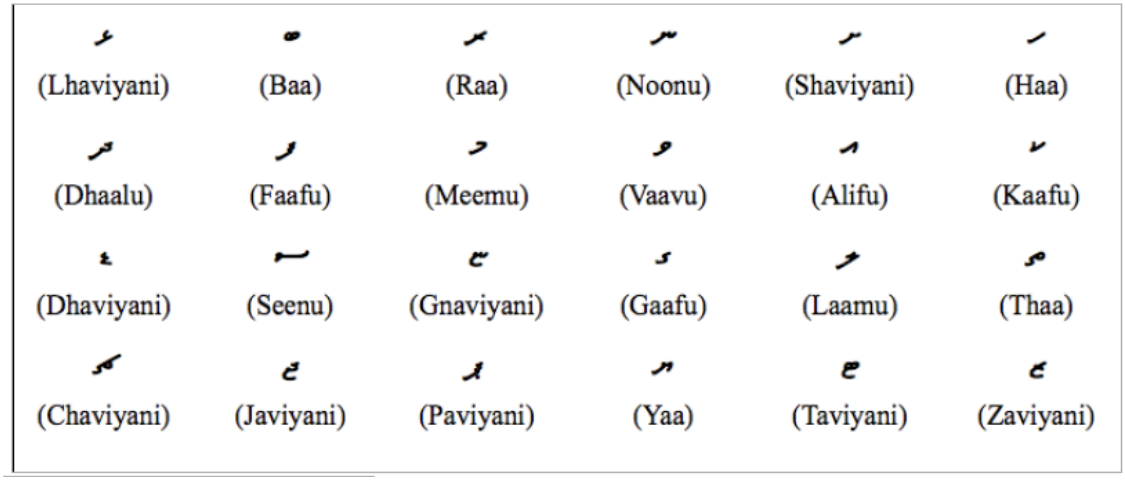


***Figure 1**: Basic Consonants*

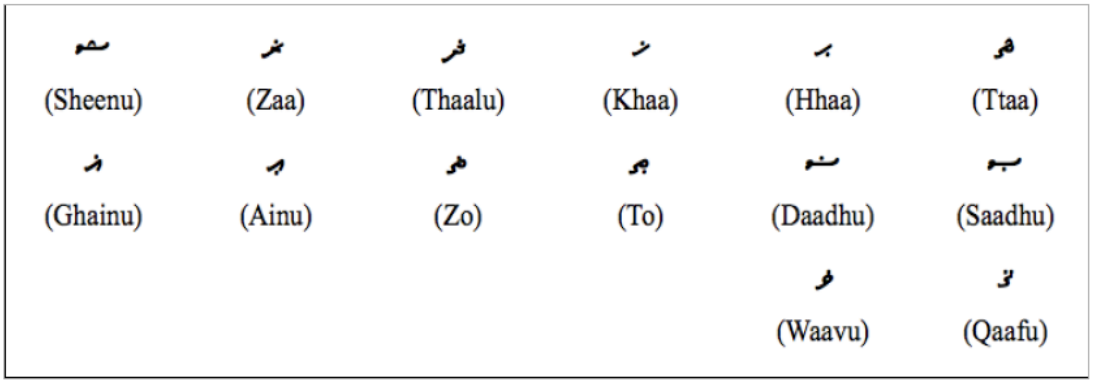


***Figure 2:** Arabic Extensions*

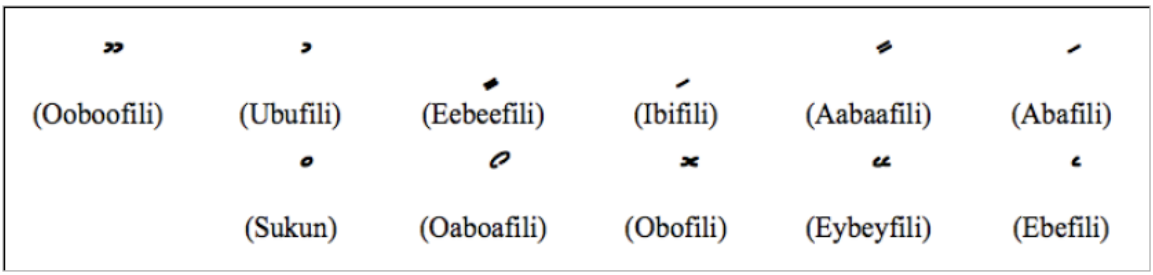


***Figure 3:** Vowels*

Figure 4: *Image of Dhivehi text showing baseline and Consonant/Vowel combinations*

2.4 Tesseract OCR Engine

Initially developed by Hewlett-Packard (HP) between 19984 and 1994 (Smith, 2007), the Tesseract OCR Engine is now an open-source OCR engine maintained by Google ("tesseract-ocr", n.d.). It is one of the most accurate OCR engines freely available today and was among the top in the 4th UNLV Annual Test of OCR Accuracy in 1995 (Rice, Jenkins, & Nartker, 1995). Since then, it has gone through major changes and improvements. In addition to the typical Latin derived character sets, it is now able to recognise more complex multilingual scripts (Smith, Antonova & Lee, 2009).

The Tesseract OCR Engine is designed to be first trained (Smith, 2007) for the target language before it can perform OCR. Once the language data has been generated, it can be portably used by anyone wishing to OCR text based on that language. There are already several language data files ("Downloads – tesseract-ocr", n.d.) that have been generated by others. Unfortunately, Dhivehi is not one of these. The next section outlines the methodology used to train Tesseract to generate the language data file for Dhivehi.

The latest version available at the time of writing this paper was 3.02. However, version 3.01 has been used for the purpose of this paper.

3. Methodology

The approach used in conducting the experiment to perform Dhivehi OCR using the Tesseract OCR Engine was split into two phases; 1) *Training Phase*: training of the Tesseract OCR Engine to generate the language data for Dhivehi; and 2) *Testing Phase*: testing the performance of Dhivehi OCR using the language data.

3.1 Training Phase

The training was conducted according to the guide on training Tesseract for a new language ("TrainingTesseract3- tesseract-ocr", 2013). All mandatory steps were followed as per this guide. The following paragraphs these steps in more detail. As per the training guide, the country code *div* was used throughout the training phases as it is the *ISO 639-3* code for Dhivehi.

3.1.1 Preparing the training data

The training character set only consisted of the basic consonants (Figure 1) and vowels (Figure 3). With the exclusion of the Arabic extensions (Figure 2), numerals and other special characters, the remaining characters were still sufficient for the purpose of this experiment. Therefore, a total of 276 character images were generated, which included a combination of every vowel with each consonant, and each consonant by itself. The consonant-vowel pairs were chosen as individual training elements due the similarities in shape between some vowels and consonants. Figure 5 shows an extract from the training set.

Figure 5: *Training Set*

The training images were generated using PIL library in Python. The characters were formatted using Faruma, which is the most widely used unicode font for Dhivehi. During this step, the size of the font was crucial for properly identifying the bounding boxes of the characters. In order to determine the most suitable font size, a series of the same images were generated using varying font sizes. By performing the next step, it was then determined that size 20 of the Faruma font was most suitable.

3.1.2 Preparing the box file

The purpose of generating a box file was basically to identify the boundary of each character, hence the term box. A number of issues were identified during this step.

Smaller font sizes between 12 and 18 produced the least number of boxes. However, larger font sizes identified consonants and their vowels as separate characters instead of a pair (Figure 6). Vowels are similar to diacritics that appear over or below the consonants, hence must be treated as a single unit.

Figure 4: *Consonant-Vowel Separate*

Figure 4: *Consonant-Vowel Pair*

To remedy the issue of the consonant and vowel being separate, the box file generation command was re-executed with the additional configuration parameter textord_min_linesize; by changing its default value from 1.25 to 2.5; and also with page segmentation mode (psm) as 6. These configuration options were consistently used throughout the training phases.

Once the box file was generated, a box file editor was used to correctly label each box with the corresponding consonant or consonant-vowel unicode characters. This can also be done directly on the box file; however, there are a number of tools available which can make this process much simpler.

3.1.3 Preparing the feature file

The feature file *.tr was then generated using the same configuration options used in the second step. This was done by invoking the training mode in Tesseract.

3.1.4 Preparing the character set

The character set indicates all possible characters Tesseract can output. This was done using the box file which was previously generated. This resulted in the unicharset file.

3.1.5 Preparing the font style information

The font style information, font_properties file, tells Tesseract the styles supported by the trained font in the format <fontname><italic><bold><fixed><serif><fraktur>. Since Faruma does not have any of these, all switches were set as 0.

3.1.6 Preparing the shape features' clusters

3.1.7 Preparing the language data file

During the final step, the inttemp, pffmtable and normproto files were prefixed with the div country code. Finally all the generated files were then combined to create the language data file, div.traineddata, for Dhivehi.

3.2 Testing Phase

Following the training, the language data file was then used to perform OCR. However, a separate test dataset was created to measure its performance. This dataset was created using the same approach used to create the training data. The rationale was to ascertain any shortcomings in the initial training approach, especially since there are no previous benchmarks for Dhivehi OCR.

Based on this understanding, the previously considered consonant-vowel pairs were then used to generate images of random Dhivehi text. With machine-generated text, one significant advantage was to eliminate certain pre-processing tasks such as noise elimination, skew detection and correction. This would not be the case in a real world scenario; however, this enabled a more accurate assessment of Dhivehi OCR performance using Tesseract.

4. Findings

A total of 239 consonant-vowel pairs were tested and it was found that the Tesseract OCR Engine had an accuracy of 69.46% for recognising machine-generated Dhivehi text. From these tests, two important observations could be made.

Firstly, the correct identifications and the overall accuracy show great potential for using Tesseract as a Dhivehi OCR tool. The following table (Table 1) is an example of successful recognition by Tesseract.

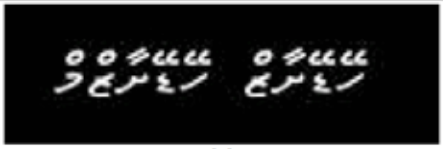
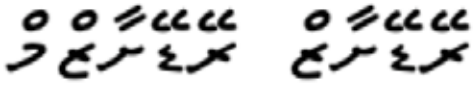
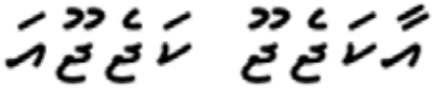
Test Image	OCR Text
 (a)	 (b)
 (c)	 (d)

Table 1: *Successful identification of Dhivehi text*

Secondly, Tesseract has mistaken the vowels on top and below the consonants as different lines, and hence tried to identify them as consonants. As can be seen in Table 1 (b) and (d), the bottom line matches the consonants in the test images (a) and (c) respectively. However, the top lines were falsely recognised as consonants, displaying as extra lines of consonants. This issue is discussed in greater detail in the next section.

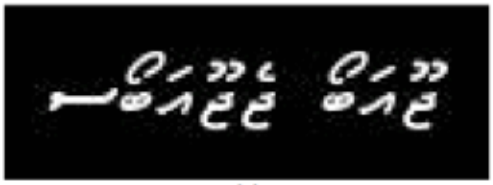
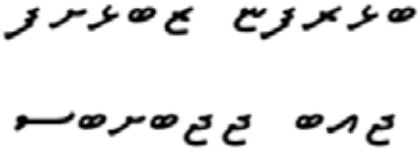
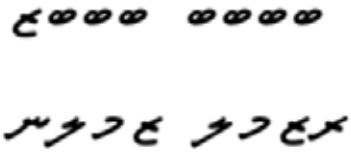
Test Image	OCR Text
 (a)	 (b)
 (c)	 (d)

Table 2: *False identifications of Dhivehi text*

5. Discussion

Based on the observations made, it can be said that Dhivehi OCR is possible without reinventing the wheel by further developing existing technologies such as Tesseract. However, despite the significant improvements currently made to support multilingual scripts, there are still critical limitations in its design. This is because Tesseract, at its core, was designed to identify English text; hence every new language introduced to it brings new challenges. The same is true for Dhivehi as well.

In Dhivehi, the baseline of the letters is at the upper bounds of the majority of consonants in Thaana script. In English, the baseline is where most of the characters sit, but in Dhivehi it is where all consonants are hung. Unlike other right-to-left languages such as Arabic, Dhivehi cannot be written without vowels, because they are not just diacritics, but are semantically essential components in forming words. Thus, it is absolutely necessary to be able to recognise the vowels as well as the consonants.

Based on these observations, the most significant errors were the result of failure to identify the consonant-vowel pairs. This is because vowels are sometimes treated as a separate line during segmentation due to the nature in which vowels are aligned and spaced out on the top and bottom of the consonants. By altering the configuration parameters, these errors were reduced but not eliminated. With a lack of detailed documentation of Tesseract configuration parameters and its design, it would be premature to deem this as a limitation of the Tesseract OCR Engine. However, it is clear that more research needs to be done on improving this aspect to have a practical level of accuracy. Minor errors such as false recognition of one or two consonants or vowels can be significantly reduced by the integration of other language tools such as word lists and dictionaries.

The objective of this paper was not to create a production-level language data file that can be used practically for identifying Dhivehi text, but to explore the suitability of a freely-available OCR Engine, Tesseract, so that its potential and limitations could be explored. With this in mind, the Tesseract OCR Engine was trained to a bare minimum without extra enhancements already available in Tesseract, such as Dictionary Data and character ambiguity specifier – unicharambigs ("TrainingTesseract3 – tesseract-ocr", 2013). Further investigations need to be done to determine the performance of integrating these options.

6. Conclusion

There is a significant gap in published literature with regard to Dhivehi OCR. This paper has been a small contribution to the vast body of OCR knowledge, but within the context of the Dhivehi language. As it is evident from the findings, the Tesseract OCR Engine offers promising results in accurately recognising Thaana script; however, there are critical problems that are script-specific that must be researched in-depth to find viable solutions. One significant area for potential research is in solving the problem of segmentation of consonants and vowels in individual lines. The training and test data were very much constrained to control external influences, hence generating them via code. However, the plethora of problems that is yet to be discovered when processing scanned documents of varying fonts or even handwritten text are yet to be discovered. Thus, future research will be done by expanding the character set to include Arabic extensions and scanned documents.

7. Acknowledgements

The author would like to thank Google and all the contributors for continuing support for the Tesseract OCR Engine, without whom this paper would not be possible. The author would also like to thank future researchers of Dhivehi language technologies, who are much needed for the development and sustenance of the Dhivehi language.

References

Cheng-Lin, L. Jaeger, S., & Nakagawa, M. (2004). Online recognition of Chinese characters: The state-of-the-art. IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(2), 198-213. doi: 10.1109/TPAMI.2004.1262182

Eikvil, L. (1993). Optical Character Recognition. http://citeseer.ist.psu.edu/142042.html.

Elagouni, K., Garcia, C., Mamalet, F., &Sébillot, P. (2013). Text recognition in multimedia documents: a study of two neural-based OCRs using and avoiding character segmentation. International Journal on Document Analysis and Recognition (IJDAR), 1-13. doi: 10.1007/s10032-013-0202-7

Fritz, S. (2002). The Dhivehi Language: a descriptive and historical grammar of Maldivian and its dialects: Ergon-Verlag.

Gair, J. W. (2010). Concise Encyclopedia of Languages of the World (K. Brown & S. Ogilvie Eds.): Access Online via Elsevier.

Google.(2013). TrainingTesseract3 - tesseract-ocr. 2013. Retrieved from http://code.google.com/p/tesseract-ocr/wiki/TrainingTesseract3

Google.(n.d.).tesseract-ocr. 2013. Retrieved from http://code.google.com/p/tesseract-ocr/

Google.(n.d.). Downloads - tesseract-ocr. 2013, Retrieved from http://code.google.com/p/tesseract-ocr/downloads/list

Handel, P. W. (1933). U.S. Patent 1915993.

Kae, A., Smith, D. A., & Learned-Miller, E. (2011). Learning on the fly: A font-free approach toward multilingual OCR. International Journal on Document Analysis and Recognition (IJDAR), 14(3), 289-301. doi: 10.1007/s10032-011-0164-6

Mori, S., Nishida, H., & Yamada, H. (1999). Optical Character Recognition: John Wiley & Sons, Inc.

Mori, S., Suen, C. Y., & Yamamoto, K. (1992).Historical review of OCR research and development. Proceedings of the IEEE, 80(7), 1029-1058. doi: 10.1109/5.156468

Peng, X., Cao, H., Setlur, S., Govindaraju, V., & Natarajan, P. (2013).Multilingual OCR research and applications: an overview. Paper presented at the Proceedings of the 4th International Workshop on Multilingual OCR, Washington, D.C.

Rice, S. V., Jenkins, F. R., & Nartker, T. A. (1995).The fourth annual test of OCR accuracy.1995 Annual Report of ISRI, University of Nevada, Las Vegas, 11-50.

Smith, R. (2007). An overview of the Tesseract OCR engine. Paper presented at the Ninth International Conference on Document Analysis and Recognition, 2007. ICDAR 2007.

Smith, R., Antonova, D., & Lee, D. S. (2009).Adapting the Tesseract open source OCR engine for multilingual OCR. Paper presented at the Proceedings of the International Workshop on Multilingual OCR, Barcelona, Spain.

Smith, R. W. (2009). Hybrid page layout analysis via tab-stop detection. Paper presented at the 10th International Conference on Document Analysis and Recognition, 2009. ICDAR'09.

Tauschek, G. (1935). U.S. Patent 2026329.