Proceedings of the Workshop on Computational Methods for Endangered Languages

Volume 1 *Proceedings of the 3rd Workshop on Computational Methods for Endangered Languages Vol. 1 Papers*

Article 4

2-26-2019

OCR Evaluation Tools for the 21st Century

Eddie A. Santos
National Research Council Canada, University of Alberta, Eddie.Santos@nrc-cnrc.gc.ca

OCR evaluation tools for the 21st century

Eddie Antonio Santos

Canadian Indigenous languages technology project, National Research Council Canada
Alberta Language Technology Lab, University of Alberta, Edmonton, Canada
Eddie.Santos@nrc-cnrc.gc.ca

Abstract

We introduce `ocreval`, a port of the ISRI OCR Evaluation Tools, now with Unicode support. We describe how we upgraded the ISRI OCR Evaluation Tools to support modern text processing tasks. `ocreval` supports producing character-level and word-level accuracy reports, supporting all characters representable in the UTF-8 character encoding scheme. In addition, we have implemented the Unicode default word boundary specification in order to support word-level accuracy reports for a broad range of writing systems. We argue that character-level and word-level accuracy reports produce confusion matrices that are useful for tasks beyond OCR evaluation—including tasks supporting the study and computational modeling of endangered languages.

1 Introduction

Optical Character Recognition (OCR) is the process of detecting text in a image, such as a scan of a page, and converting the written language into a computer-friendly text format. While OCR is well-established for majority languages (Tesseract Langdata), the same cannot be said for endangered languages (Hubert et al., 2016). When developers create OCR models for low-resources languages, they must have a tool for evaluating the accuracy of their models. This will aid in making decisions on how to model the language, and how to tweak parameters in order to accurately recognize text in that language, especially when labelled materials are scarce.

A further complication encountered when developing OCR models for endangered languages is that of orthography—a language may have *many* orthographies, or may not have a standardized orthography at all. And if any orthographies exist, they likely contain characters beyond the non-accented Latin alphabet. IPA-influenced glottal stop (ʔ) and schwa (ə) characters have a tendency to sneak into new orthographies. Historically, computer software provides poor support for characters outside the range of those characters commonly found in Western European majority language orthographies.

The ISRI OCR Evaluation Tools were last released in 1996 by the University of Nevada, Las Vegas's Information Science Research Institute (Rice and Nartker, 1996). Despite their age, the tools continue to be useful to this day for the task of evaluating new OCR models (Hubert et al., 2016). Its `accuracy` and `wordacc` tools are of particular utility. However, being somewhat dated, it has limited and awkward support for non-European orthographies. Since the tools' 1996 release, the Unicode standard has gradually improved support for non-European characters, introducing many new scripts and characters. However, the ISRI OCR Evaluation Tools have not been upgraded since.

In this paper, we present an updated version of the ISRI OCR Evaluation Tools, which we named `ocreval`. Our contributions include

- Port the ISRI OCR Evaluation Tools onto modern macOS and Ubuntu Linux systems;
- Add support for UTF-8, enabling the analysis of 1,112,064 unique characters;
- Implement the default Unicode word boundary specification to evaluate word accuracy.

We describe in more depth what we have changed and why; and we postulate how `ocreval` can be used to support **tasks unrelated to OCR** that may benefit people working in endangered language study and tool development.

Before continuing, we would be remiss not to mention OCRevalUAtion (Carrasco, 2014), an alternative to the toolkit presented in this paper.

Proceedings of the 3rd Workshop on the Use of Computational Methods in the Study of Endangered Languages: Vol. 1 Papers, pages 23–27,
Honolulu, Hawai'i, USA, February 26–27, 2019.

2 How does `ocreval` extend the ISRI OCR Evaluation Tools?

`ocreval` is a port of the ISRI OCR Evaluation Tools to modern systems. This means that much of the codebase and functionality is the same as the codebase written in 1996; however, the original code presents many incompatibilities with current systems (usage of older programming language features and reliance on out-of-date source code libraries). When we downloaded the ISRI OCR Evaluation Tools for the first time, its C programming language source code would not compile (translate source code into a usable application). After a few modifications to the source code, we were able to compile and run the ISRI OCR Evaluation Tools on our systems.

The second problem is how text was represented in the system. The ISRI OCR Evaluation Tools were written in the early days of Unicode, when Unicode was strictly a 16-bit character set. As such, the ISRI OCR Evaluation Tools can handle the first 65,536 characters defined in the Unicode standard; however, as of Unicode 11.0 (Unicode Consortium, 2018), there are 1,112,064 possible characters total.[1] Only 137,374 (12.33%) characters are defined in the Unicode standard, with the rest of the code space reserved for future use. This means that the ISRI OCR Evaluation Tools can only represent half of the characters defined in the modern day Unicode standard. Among the characters that are not representable in the ISRI OCR Evaluation Tools are characters in recently created writing systems, such as the Osage alphabet (Everson et al., 2014), as well as historical writing systems (Cuneiform, Linear B, Old Italic). Thus, `ocreval` extends the ISRI OCR Evaluation Tools by internally representing characters using the UTF-32 character encoding form, allowing for the analysis of characters from writing systems both new and old.

2.1 UTF-8 support

The most significant hurdle remaining is how Unicode characters are input and output into `ocreval`. As of October 2018, UTF-8 is the most common character encoding scheme on the web, used for 92.4% of websites (W3Techs). This is not surprising, as UTF-8 represents *all* 1,112,064 characters possible in the Unicode standard in a way that is backwards-compatible with 7-bit ASCII. The ISRI OCR Evaluation Tools predate the widespread adoption of UTF-8; as such, it assumes that any text input is encoded in ISO-8859-1 or "Latin-1" encoding. Latin-1, intended for use with Western European languages, only encodes about 192 printable characters, most of which are Latin alphabetic characters; as such, Latin-1 is usually inadequate for encoding endangered and minority languages. The ISRI OCR Evaluation Tools *does* support 16-bit Unicode (65,536 characters), however, it uses an *ad hoc* format called "Extended ASCII". The ISRI OCR Evaluation Tools bundled the `uni2asc` and `asc2uni` tools to convert to and from the Extended ASCII format and the (now outdated) 16-bit UCS-2 character encoding. Anybody wishing to use characters outside the range of Latin-1 would have to first use `uni2asc` before using their documents with any of the available tools. If their original documents were encoded in UTF-8, they would have to do two conversions: first, convert from UTF-8 to UCS-2 using a tool such as `iconv`; then, convert the UCS-2 file into Extended ASCII using `uni2asc`.

In a time when UTF-8 exists and is readily available, we find that Extended ASCII is too much of a hassle. As such, we modified the file reading and writing routines used in all utilities provided in `ocreval` to open files in the UTF-8 character encoding scheme exclusively, obviating the need to convert into an *ad hoc* format such as Extended ASCII. Since all input and output is done in UTF-8, we removed the now-redundant `asc2uni` and `uni2asc` utility programs from `ocreval`.

2.2 Unicode word segmentation

One of the ISRI OCR Evaluation Tools's most useful utilities is `wordacc`. As its name implies, `wordacc` computes the recognition accuracy for entire words, rather than for single characters alone. However, what constituted as a "word" was rather narrowly defined in the previous version. Originally, a "word" was a series of one or more consecutive characters in the range of lowercase ASCII characters (U+0061–U007A), or lowercase Latin-1 characters (U+00DF–U+00F6, U+00F8–U+00FF). Input would have to be converted to lowercase before calculating word accuracy. Any other characters were *not* considered to be part of a word, and hence, would not appear in word accu-

[1] There are 1,114,112 values defined in the Unicode 11.0 code space, however 2048 of these are *surrogate characters* which cannot encode a character by themselves; hence, we removed these characters from the total.

racy summaries or confusion matrices. This narrow definition of a "word" limits the usefulness of `wordacc`, even for Latin-derived scripts.

To broaden its usefulness, we changed how `ocreval` finds words in text by adopting Unicode's default word boundary specification (Unicode Standard Annex #29). This specifies a reasonable default for finding the *boundaries* between words and other words, and between words and non-word characters. Then, to find words proper, we extract character sequences between boundaries that start with a "word-like" character, such as letters, syllables, digits, and symbols that frequently appear as parts of words. We also considered *private-use characters* to be "word-like" characters so that if a language's orthography is not yet encoded or is poorly-supported in Unicode, its characters can be represented using private-use characters—which are set aside by the Unicode standard as allocated code points to represent any character the user requires.

A caveat is that this algorithm will *not* work for all writing systems. Scripts like Thai, Lao, and Khmer, which do not typically have any spaces separating words, will not be handled properly. As such, this is not a "one-size-fits-all" solution, and may need to be tailored depending on the language.

2.3 Installing `ocreval`

In all cases, the user must have a basic understanding of the command line interface of their computer. However, we have tried to document and streamline the process when possible.

`ocreval` can be installed on a modern macOS system using the Homebrew package manager:

```
$ brew tap eddieantonio/eddieantonio
$ brew install ocreval
```

On Ubuntu Linux, the tools can be installed from its source code. After downloading the zip archive,[2] install all system dependencies and extract the contents of the source code archive. Within the newly created directory, issue the `make` command from the command line:

```
$ sudo apt update
$ sudo apt install build-essential \
    libutf8proc-dev unzip
$ unzip ocreval-master.zip
$ cd ocreval-master/
$ make
$ sudo make install
```

[2] https://github.com/eddieantonio/ocreval/archive/master.zip

`ocreval` can also be installed on Windows 10 within the Ubuntu app, obtainable in the Microsoft app store. Copy the downloaded zip archive into the Ubuntu subsystem, then follow the same steps as the Ubuntu Linux install.

2.4 The importance of open source

Software is **open source** when its source code is publicly-accessible, under a license that permits anyone to make modifications and share the modifications with others. We have released `ocreval` as open source for many reasons: it maximizes the amount of people that can benefit from the tool by making it freely-accessible; hosting the software on the collaborative coding platform GitHub allows for people around the world to share enhancements to `ocreval` for everybody's benefit; and the ISRI OCR Evaluation Tools were originally released as open source.

`ocreval` is maintained on GitHub at `https://github.com/eddieantonio/ocreval`. On GitHub, changes to the source code are transparent and publicly-visible. Contributions are welcome in the form of **issue reports** and **pull requests**. Issue reports alert us to any bugs or inadequacies found in the currently published version of the software; pull requests allow volunteers to write suggested source code enhancements to share with the rest of the community. Creating an issue report or a pull request both require a GitHub account. We welcome anyone who wants to contribute to join in; no contribution is too small!

The ISRI OCR Evaluation Tools were released under an open source licence. The significance of this cannot be overstated. If the original source code was not available on (the now defunct) Google Code repository, this paper would not have been possible. As such, our contributions are also released under an open source license, in the hopes that it may also be useful in 20 years time.

3 How can `ocreval` help endangered languages?

Hubert et al. (2016) have already used `ocreval` to evaluate the accuracy of OCR models for Northern Haida, a critically-endangered language spoken in Western Canada. Unicode support was integral to representing and evaluating the idiosyncratic orthography specific to the corpus in question (which differs from modern-day Haida orthographies).

Previously, we mentioned that among the ISRI

OCR Evaluation Tools' most useful utilities are `accuracy` and `wordacc`. We see these utilities in `ocreval` to be much more general-purpose tools, not only limited to the evaluation of OCR output. Fundamentally, these tools compare two text files, producing a report of which character sequences are misclassified or "confused" for other character sequences. This concept is useful beyond the evaluation of OCR models.

One possible non-OCR application could be to study language change. Parallel texts—whether it be prose, or a word list—could be fed into `accuracy`. One part of the input pair would be representative of language spoken or written in the modern day, and the other part of the input would be a historical example, reconstructed text, or text from a distinct dialect. `accuracy` will then produce minimal "confusions", or sequences of commonly misclassified sequences. Since `accuracy` prints statistics for how often a confusion is found, `accuracy` inadvertently becomes a tool for reporting systematic changes, along with how often the effect is attested. Preliminary work by Arppe et al. (2018) used `ocreval` to compare Proto-Algonquian to contemporary Plains Cree. Using an extensive database with over ten thousand pairings of morphologically simple and complex word stems—mapping each modern Cree word form with the corresponding reconstructed Proto-Algonquian form—they found that the contents of the confusion matrix matched quite closely with the posited historical sound change rules. In addition, the confusion matrix can be used to quantify how often a particular rule applies in the overall Cree vocabulary. The benefit of `ocreval`'s added Unicode support facilitates this use case, as sound change rules are hardly ever representable in only Latin-1 characters.

Lastly, throughout this paper we have made the assumption that corpora are encoded in UTF-8—thus, using `ocreval` should be straightforward. However, this is not always the case for minority language resources, even if they are encoded digitally. One way minority language texts may have been encoded is by "font-encoding" or "font-hacking". This is a specially-designed font *overrides* the display of existing code points, as opposed to using existing Unicode code points assigned for that particular orthography. This may be because the font was defined for pre-Unicode systems, or because Unicode lacked appropriate character assignments at the development of said font. For example, in place of the ordinary character for "©", the font will render "ğ", and in place of "¬", the font will render "Í". For these cases, `ocreval` alone is insufficient; an external tool must be used such as `convertextract` (Pine and Turin, 2018).

4 Conclusion

We presented `ocreval`, an updated version of the ISRI OCR Evaluation Tools. `ocreval` provides a suite of tools for evaluating the accuracy of optical character recognition (OCR) models—however, we postulate that the evaluation tools can be generalized to support other tasks. Our contributions include porting the old codebase such that it works on modern systems; adding support for the ubiquitous UTF-8 character encoding scheme, as well internally representing characters using the full Unicode code space; and the implementation of the default Unicode word boundary specification, which facilitates finding words in a variety of non-Latin texts. We released `ocreval` online as open-source software, with the intent to make our work freely-accessible, as well as to encourage contributions from the language technology community at large.

Acknowledgments

The author wishes to thank Isabell Hubert for involving us in this project; the author is grateful to Antti Arppe, Anna Kazantseva, and Jessica Santos for reviewing early drafts of this paper.

References

Antti Arppe, Katherine Schmirler, Eddie Antonio Santos, and Arok Wolvengrey. 2018. Towards a systematic and quantitative identification of historical sound change rules — Algonquian historical linguistics meets optical character recognition evaluation. Working paper presented in Alberta Language Technology Lab seminar, November 19, 2018. http://altlab.artsrn.ualberta.ca/wp-content/uploads/2019/01/OCR_and_proto-Algonquian_nonanon.pdf.

Rafael C Carrasco. 2014. An open-source OCR evaluation tool. In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*, pages 179–184. ACM. https://github.com/impactcentre/ocrevalUAtion.

Michael Everson, Herman Mongrain Lookout, and Cameron Pratt. 2014. Final proposal to encode the

Osage script in the UCS. Technical report, ISO/IEC JTC1/SC2/WG2. Document N4619.

Homebrew. 2009. Homebrew: The missing package manager for macOS. https://brew.sh/. (Accessed on 10/22/2018).

Isabell Hubert, Antti Arppe, Jordan Lachler, and Eddie Antonio Santos. 2016. Training & quality assessment of an optical character recognition model for Northern Haida. In *LREC*.

Aidan Pine and Mark Turin. 2018. Seeing the Heiltsuk orthography from font encoding through to Unicode: A case study using convertextract. *Sustaining Knowledge Diversity in the Digital Age*, page 27.

Stephen V Rice and Thomas A Nartker. 1996. The ISRI analytic tools for OCR evaluation. *UNLV/Information Science Research Institute, TR-96-02*.

Tesseract Langdata. 2018. tesseract-ocr/langdata: Source training data for Tesseract for lots of languages. https://github.com/tesseract-ocr/langdata. (Accessed on 10/22/2018).

The Unicode Consortium, editor. 2018. *The Unicode Standard, Version 11.0.0*. The Unicode Consortium, Mountain View, CA.

Unicode Standard Annex #29. 2018. Unicode text segmentation. Edited by Mark Davis. An integral part of The Unicode Standard. http://unicode.org/reports/tr29/. (Accessed on 10/22/2018).

W3Techs. 2018. Usage statistics of character encodings for websites, october 2018. https://w3techs.com/technologies/overview/character_encoding/all. (Accessed on 10/22/2018).