Proceedings of 1st Workshop on Language Technologies for Historical and Ancient Languages, pages 47–51
Language Resources and Evaluation Conference (LREC 2020), Marseille, 11–16 May 2020

A Tool for Facilitating OCR Postediting in Historical Documents

Alberto Poncelas, Mohammad Aboomar, Jan Buts, James Hadley, Andy Way
ADAPT Centre, School of Computing, Dublin City University, Ireland
Trinity Centre for Literary and Cultural Translation, Trinity College Dublin, Ireland
{alberto.poncelas, jan.buts andy.way}@adaptcentre.ie
{ABOOMARM, HADLEYJ}@tcd.ie

Abstract

Optical character recognition (OCR) for historical documents is a complex procedure subject to a unique set of material issues, including inconsistencies in typefaces and low quality scanning. Consequently, even the most sophisticated OCR engines produce errors. This paper reports on a tool built for postediting the output of Tesseract, more specifically for correcting common errors in digitized historical documents. The proposed tool suggests alternatives for word forms not found in a specified vocabulary. The assumed error is replaced by a presumably correct alternative in the post-edition based on the scores of a Language Model (LM). The tool is tested on a chapter of the book *An Essay Towards Regulating the Trade and Employing the Poor of this Kingdom* (Cary, 1719). As demonstrated below, the tool is successful in correcting a number of common errors. If sometimes unreliable, it is also transparent and subject to human intervention.

Keywords: OCR Correction, Historical Text, NLP Tools

1. Introduction

Historical documents are conventionally preserved in physical libraries, and increasingly made available through digital databases. This transition, however, usually involves storing the information concerned as images. In order to correctly process the data contained in these images, they need to be converted into machine-readable characters. This process is known as optical character recognition (OCR). Converting a book from image into text has obvious benefits regarding the identification, storage and retrieval of information. However, applying OCR usually generates noise, misspelled words and wrongly recognised characters. It is therefore often necessary to manually postedit the text after it has undergone the automatic OCR process. Usually, the errors introduced by the OCR tool increase with the age of the document itself, as older documents tend to be in worse physical condition. The circumstances of digitization, e.g. the quality of the scan and the mechanical typeset used, also impact the outcome of the OCR procedure. This paper proposes a tool for automatically correcting the majority of errors generated by an OCR tool. String-based similarities are used to find alternative words for perceived errors, and a Language Model (LM) is used to evaluate sentences. This tool has been made publicly available.[1]

The performance of the tool is evaluated by correcting the text generated when using OCR with the book *An Essay Towards Regulating the Trade and Employing the Poor of this Kingdom* (Cary, 1719).

2. Related Work

To improve the outcome of OCR, one can either focus on the processing of images in the scanned book, or on editing the output of the OCR tool. For either stage, several approaches have been proposed.

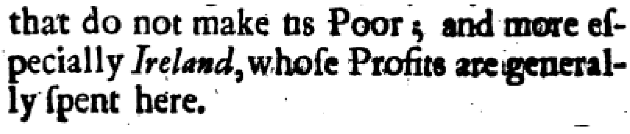
The approaches involving image-processing perform modifications on the scanned book that make the OCR perform better. Examples of these approaches include adding noise, as through rotation, for augmenting the training set (Bieniecki et al., 2007), reconstructing the image of documents in poor condition (Maekawa et al., 2019), clustering similar words so they are processed together (Kluzner et al., 2009) or jointly modeling the text of the document and the process of rendering glyphs (Berg-Kirkpatrick et al., 2013).

Techniques for increasing accuracy by performing post-OCR corrections can be divided into three sub-groups. The first group involves lexical error correction, and consists of spell-checking the OCR output using dictionaries, online spell-checking (Bassil and Alwani, 2012), and using rule-based systems for correcting noise (Thompson et al., 2015). The second group of strategies for correcting OCR output is context-based error correction, in which the goal is to evaluate the likelihood that a sentence has been produced by a native speaker by using an n-gram LM to evaluate the texts produced by the OCR (Zhuang et al., 2004), and to use a noisy-channel model (Brill and Moore, 2000), or a Statistical Machine Translation engine (Afli et al., 2016) to correct the output of the OCR. A final approach proposes using several OCR tools and retrieving the text that is most accurate (Volk et al., 2010; Schäfer and Weitz, 2012).

3. OCR Challenges for Historical Document

Performing OCR is a challenging task. Although ideally the procedure should successfully generate the text represented in an image, in practice the tools often produce errors (Lopresti, 2009). In addition, when older documents are converted into text further difficulties arise that cause the performance of the OCR tools to decrease. One of the problems of historical documents is that the quality of the print medium has often degraded over time. The quality of the paper also impacts the output, as in some cases the letters on the reverse side of a page are visible in the scanned image, which adds noise to the document.

[1] https://github.com/alberto-poncelas/tesseract_postprocess



that do not make us Poor ; and more eſ-
pecially *Ireland*, whoſe Profits are general-
ly ſpent here.

Figure 1: Example of the scan of the book *An Essay Towards Regulating the Trade*.

Furthermore, OCR systems are generally best suited to contemporary texts, and not built to handle the typefaces and linguistic conventions that characterize older documents. In Figure 1 we show a small extract of a scan of the book *An Essay Towards Regulating the Trade* to illustrate some of the problems frequently encountered. One may notice the following particularities of the text:

- Some words such as *especially*, *whose* and *spent* contain the "ſ" or long "s", an archaic form of the letter "s" which can easily be confused with the conventional symbol for the letter "f".
- Some words, such as *Poor* and *Profits*, are capitalized even though they occur mid-sentence. This would be unusual in present-day English.

Piotrowski (2012) categorizes variations in spelling as uncertainty (digitization errors), variance (inconsistent spelling) and difference (spelling that differs from contemporary orthography). In our work we focus on the latter.
Spelling issues compound the general challenges touched upon before, such as the quality of the scan (e.g. the word "us" in Figure 1 is difficult to read even for humans). Further issues include the split at the end of the line (e.g. the word "especially" or "generally").

4. Proposed Tool

This paper introduces a tool that automatically edits the main errors in the output of an OCR engine, including those described in Section 3.. The method retains, next to the edited text, the material that has been automatically replaced. Thus, the human posteditor has the agency to approve or discard the changes introduced by the tool. The aim of automating the initial replacement procedure is to shorten the overall time spent on post-editing historical documents.
In order to execute the tool, run the command `ocr_and_postprocess.sh $INPUT_PDF $OUT $INITPAGE $ENDPAGE`. In this command, `$INPUT_PDF` contains the path of the *pdf* file on which OCR will be performed, and `$OUT` the file where the output will be written. `$INITPAGE` and `$ENDPAGE` indicate from which page until which page the OCR should be executed.
The output is a file consisting of two columns (tab-separated). The first column contains the text after OCR is applied and the errors have been corrected. In the second column, we include the list of edits performed by our tool, so that a human post-editor can easily identify which words have been replaced.
The pipeline of this approach is divided into three steps, as further explained in the subsections below. First, the OCR is executed (Section 4.1.). Subsequently, words that are unlikely to exist in English are identified and replacement words are sought (Section 4.2.). Finally, the word-alternatives are evaluated within the larger sentence in order to select the best alternative (Section 4.3.).

4.1. Perform OCR

The first step is to extract part of the *pdf* and convert it into a list of *png* images (one image per page). These images are fed to an OCR engine and thus converted into text. The line-format in the text will conform to the shape of the image, meaning that word forms at the end of a line ending on an "-" symbol need to be joined to their complementary part on the following line to ensure completeness.

4.2. Get alternative words

As the output of the OCR tool is expected to contain errors, this text is compared to a list of English vocabulary referred to as *recognized words*.
Once the text is tokenized and lowercased, some of the words can be replaced by alternatives that fit better within the context of the sentence. The words that we want to replace are those that are not included in the list of *recognized words* or contain letters that are difficult to process by the OCR tool (as in the case of confusion between the letters "ſ" and "f" mentioned in Section 3.). For each of these words we construct a list of candidates for a potential replacement. This list is built as follows:

1. Even if a word seems to be included in the list *recognized words*, it still may contain errors, as some letters are difficult for the OCR to recognize. As per the above, "ſ" can be replaced with "s", and the resultant word can be added as a replacement candidate if it is a *recognized word*.

2. If the word is not in the list *recognized words*, we proceed along the following lines:

   (a) The word is split into two subwords along each possible line of division. If both items resulting from the split are recognized words, the pair of words is added as an alternative candidate.

   (b) Similar words in the vocabulary are suggested using a string-distance metric. The 3 closest words, based on the *get_close_matches* function of python's *difflib* library, are included.

After this step, for a word w_i we have a list of potential replacements $w_i, w_i^{(1)}, w_i^{(2)}...w_i^{(r_i)}$, where r_i is the number of alternatives for w_i. Note also that the original word is included as an alternative.

4.3. Replace words with their alternatives

Once we have obtained a list of alternatives, we proceed to evaluate which of the alternatives fits best within the context of the sentence. This means that given a sentence consisting of a sequence of N words $(w_1, w_2...w_N)$, the word

w_i is substituted in the sentence with each of its replacement candidates, and a set of sentences is obtained as in (1):

$$\begin{aligned} \{(w_1 \dots w_i \dots w_N),\\ (w_1 \dots w_i^{(1)} \dots w_N),\\ \dots\\ (w_1 \dots w_i^{(r_i)} \dots w_N)\}. \end{aligned} \quad (1)$$

The perplexity of an LM trained on an English corpus is used to evaluate the probability that a sentence has been produced by a native English speaker. Given a sentence consisting of a sequence of N words as $w_1, w_2 ... w_N$, the perplexity of a language model is defined as in Equation (2):

$$PP = 2^{-\frac{1}{n} P_{LM}(w_1 ... w_N)} \quad (2)$$

Note that the LM evaluation is performed with lowercased sentences. Once the sentence with the lower perplexity has been selected, the case is reproduced, even if the word has been replaced. This is relevant in the case of capitalization conventions, as related to words such as *Poor* or *Profits* in Figure 1.

5. Experiments

5.1. Experimental Settings

In order to evaluate our proposal, we use the Tesseract[2] Tool (Smith, 2007) to apply OCR to the book *An Essay Towards Regulating the Trade*. Specifically, we convert into text a scan of the chapter *An Act for Erecting of Hospitals and Work-Houses within the City of Bristol, for the better Employing and Maintaining the Poor thereof* (pages 125 to 139).

The list of *recognized words* consists of the vocabulary of the python package nltk[3], expanded with a list of 467K words[4] (DWYL, 2019). For each word that is not included in the vocabulary list we search for the closest 3 alternatives (based on a string-distance metric).

In order to evaluate which word-alternative is the most plausible in the sentence we use a 5-gram LM built with KenLM toolkit (Heafield, 2011), trained on the Europarl-v9 corpus (Koehn, 2005).

5.2. Results

The text obtained after applying OCR consists of 576 lines. These lines are usually short, containing about 7 words per line.

In Figure 2 we show an extract of the scanned book. The text obtained after OCR is given in Table 1 (in the first column). Comparing the resultant text with the original, one can easily spot errors mentioned in Section 3.,such as retrieving “fuch” instead of “such”, and further irregularities, such as interpreting “time as” as a single word.

[2] https://github.com/tesseract-ocr/tesseract

[3] https://www.nltk.org/

[4] https://github.com/dwyl/english-words/blob/master/words.zip

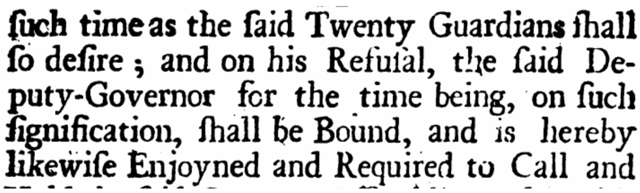

ſuch time as the ſaid Twenty Guardians ſhall
ſo deſire ; and on his Refuſal, the ſaid De-
puty-Governor for the time being, on ſuch
ſignification, ſhall be Bound, and is hereby
likewiſe Enjoyned and Required to Call and

Figure 2: Extract from the test set.

Original	Edited	Changes
{uch timeas the faid Twenty Guardians fhall	{uch times the said Twenty Guardians shall	timeas → times; faid → said; fhall → shall
fo defire ; and on his Refutal, the faid	so desire; and on his Refutal, the said	fo → so; defire → desire; faid → said
Deputy-Governor for the time being, on fuch	Ex-governor for the time being, on such	Deputy-Governor → Ex-governor; fuch → such
fignification, fhall be Bound, and is hereby	fignification, shall be Bound, and is hereby	fhall → shall
ikewife Enjoyned and Required to Call and	likewise Enjoined and Required to Call and	ikewife → likewise; Enjoyned → Enjoined

Table 1: Example of postedited line

Table 1 also presents the text after being processed with our tool (second column). In the third column we include the substitution performed (this information is also retrieved by the tool). We observe that 66% of the lines contain at least one correction. Each line has a minimum of 0 and a maximum of 3 corrections.

The tool is generally successful in correcting the words in which the letter “f” and “f” were previously confused. Most frequent in this regard are word-initial errors for “shall”, “so” and “said”, but word-internal mistakes, as in“desire” (see second row), are not uncommon.

In the first row we observe that the word “timeas” is not recognized as part of the vocabulary. The tool finds that the item can be split into the English words “time” and “as”. However, the tool also finds other options, and opts to render “times”, thus requiring human intervention and illustrating the necessity of transparency in the automated procedure. In the last row, a non-existent word has been corrected as “likewise” because it is similar in terms of string-distance and is plausible according to the LM.

Table 2 presents some of the words that could not be found in the vocabulary (first column) and their respective candidates for replacement. The tool replaced these words by the

Unrec. word	Alternatives
“faid”	“fai”, “f aid”, “fid”, “fa id”, “said”, “fraid”
“timeas”	“timias”, “tim eas”, “time as”, “tineas”, “ti meas”, “times”
“ikewife”	“likewise”, “ike wife”, “piewife”, “kalewife”

Table 2: Example of replacement dictionaries

most plausible alternative, employing th LM to evaluate the resulting sentence.

Despite numerous successful corrections, Table 1 also shows some of the limitations of the tool. For example, the word “fignification” has not been properly replaced by a correct alternative. Other words have been incorrectly replaced, such as “Deputy-Governor”, which now occurs as “Ex-Governor”.

In our experiments, we observe that around 63% of the errors are corrected by our tool. Most of the corrections are made in frequent words such as the word “shall” mentioned in Table 1.

6. Conclusion and Future Work

In this paper we have presented a tool to postprocess errors in the output of an OCR tool. As the problems addressed mainly pertains to historical documents, the tool was illustrated with reference to the early 18th-century text *An Essay Towards Regulating the Trade*. In order to achieve a more accurate representation of the original document than is commonly attained in image-text conversion, we constructed a system that identifies words that have potentially been incorrectly recognised and which suggests candidates for replacement. In order to select the best candidate, these alternatives are evaluated within the context of the sentence using an LM.

In this study we have manually stated which characters are misrecognized by the OCR system. In the future, we hope to develop a method for automatically identifying such characters.

We did not find large amounts of good-quality data from around 1700. Further research would benefit from LM models built on data from the same period as the test set, which could also be used to select appropriate sentences (Poncelas et al., 2016; Poncelas et al., 2017).

The tool could also be expanded to address related issues of textual organization, such as the automatic separation of side notes from a body of text. Overall, OCR technology is a fundamental factor in the dissemination of knowledge in the digital age, and to refine its output is essential.

7. Acknowledgements

The QuantiQual Project, generously funded by the Irish Research Council’s COALESCE scheme (COALESCE/2019/117).

This research has been supported by the ADAPT Centre for Digital Content Technology which is funded under the SFI Research Centres Programme (Grant 13/RC/2106).

8. Bibliographical References

Afli, H., Qiu, Z., Way, A., and Sheridan, P. (2016). Using SMT for OCR error correction of historical texts. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation*, pages 962–966, Portorož, Slovenia.

Bassil, Y. and Alwani, M. (2012). Ocr post-processing error correction algorithm using google’s online spelling suggestion. *Journal of Emerging Trends in Computing and Information Sciences*, 3(1).

Berg-Kirkpatrick, T., Durrett, G., and Klein, D. (2013). Unsupervised transcription of historical documents. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 207–217.

Bieniecki, W., Grabowski, S., and Rozenberg, W. (2007). Image preprocessing for improving ocr accuracy. In *2007 International Conference on Perspective Technologies and Methods in MEMS Design*, pages 75–80. IEEE.

Brill, E. and Moore, R. C. (2000). An improved error model for noisy channel spelling correction. In *Proceedings of the 38th annual meeting on association for computational linguistics*, pages 286–293, Hong Kong. Association for Computational Linguistics.

Heafield, K. (2011). KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland.

Kluzner, V., Tzadok, A., Shimony, Y., Walach, E., and Antonacopoulos, A. (2009). Word-based adaptive OCR for historical books. In *2009 10th International Conference on Document Analysis and Recognition*, pages 501–505.

Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. *Proceedings of the 10th Machine Translation Summit (MT Summit)*, pages 79–86.

Lopresti, D. (2009). Optical character recognition errors and their effects on natural language processing. *International Journal on Document Analysis and Recognition (IJDAR)*, 12(3):141–151.

Maekawa, K., Tomiura, Y., Fukuda, S., Ishita, E., and Uchiyama, H. (2019). Improving OCR for historical documents by modeling image distortion. In *Digital Libraries at the Crossroads of Digital Information for the Future: 21st International Conference on Asia-Pacific Digital Libraries, ICADL 2019*, volume 11853, pages 312–316, Kuala Lumpur, Malaysia.

Piotrowski, M. (2012). Natural language processing for historical texts. *Synthesis lectures on human language technologies*, 5(2):1–157.

Poncelas, A., Way, A., and Toral, A. (2016). Extending feature decay algorithms using alignment entropy. In *International Workshop on Future and Emerging Trends in Language Technology*, pages 170–182, Seville, Spain. Springer.

Poncelas, A., Maillette de Buy Wenniger, G., and Way, A. (2017). Applying n-gram alignment entropy to improve feature decay algorithms. *The Prague Bulletin of Mathematical Linguistics*, 108(1):245–256.

Schäfer, U. and Weitz, B. (2012). Combining ocr outputs for logical document structure markup: technical background to the acl 2012 contributed task. In *Proceedings of the ACL-2012 Special Workshop on Rediscovering 50 Years of Discoveries*, pages 104–109, Jeju, Republic of Korea.

Smith, R. (2007). An overview of the tesseract OCR engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 2, pages 629–633, Curitiba, Brazil.

Thompson, P., McNaught, J., and Ananiadou, S. (2015). Customised OCR correction for historical medical text. In *2015 Digital Heritage*, volume 1, pages 35–42.

Volk, M., Marek, T., and Sennrich, R. (2010). Reducing ocr errors by combining two ocr systems. *ECAI 2010*, page 61.

Zhuang, L., Bao, T., Zhu, X., Wang, C., and Naoi, S. (2004). A chinese OCR spelling check approach based on statistical language models. In *2004 IEEE International Conference on Systems, Man and Cybernetics*, volume 5, pages 4727–4732, The Hague, Netherlands.

9. Language Resource References

Cary, J. (1719). *An Essay Towards Regulating the Trade and Employing the Poor of this Kingdom: Whereunto is Added an Essay Towards Paying Off the Publick Debts*.

DWYL. (2019). `https://github.com/dwyl/english-words/blob/master/words.zip`.