Statistical learning for OCR error correction

Article *in* Information Processing & Management · November 2018

DOI: 10.1016/j.ipm.2018.06.001

Jie Mei
Dalhousie University
10 PUBLICATIONS **21** CITATIONS
SEE PROFILE

Aminul Islam
North South University
36 PUBLICATIONS **264** CITATIONS
SEE PROFILE

Abidalrahman Moh'd
Dalhousie University
34 PUBLICATIONS **227** CITATIONS
SEE PROFILE

Evangelos Milios
Dalhousie University
203 PUBLICATIONS **7,600** CITATIONS
SEE PROFILE

Information Processing and Management 54 (2018) 874–887

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

Statistical learning for OCR error correction

Jie Mei*,[a], Aminul Islam[b], Abidalrahman Moh'd[a], Yajing Wu[a], Evangelos Milios[a]

[a] Faculty of Computer Science, Dalhousie University, Halifax, NS B3H 1W5, Canada
[b] School of Computing and Informatics, University of Louisiana at Lafayette, Lafayette, LA 70503, Canada

ARTICLE INFO

Keywords:
OCR post-processing
OCR error
Error correction
Statistical learning

ABSTRACT

Modern OCR engines incorporate some form of error correction, typically based on dictionaries. However, there are still residual errors that decrease performance of natural language processing algorithms applied to OCR text. In this paper, we present a statistical learning model for post-processing OCR errors, either in a fully automatic manner or followed by minimal user interaction to further reduce error rate. Our model employs web-scale corpora and integrates a rich set of linguistic features. Through an interdependent learning pipeline, our model produces and continuously refines the error detection and suggestion of candidate corrections. Evaluated on a historical biology book with complex error patterns, our model outperforms various baseline methods in the automatic mode and shows an even greater advantage when involving minimal user interaction. Quantitative analysis of each computational step further suggests that our proposed model is well-suited for handling volatile and complex OCR error patterns, which are beyond the capabilities of error correction incorporated in OCR engines.

1. Introduction

Ongoing effort on large-scale digitization has made a massive amount of printed information available to search, access, and analyze. Besides some well-known Internet archives, such as Google Books,[1] Biodiversity Heritage Library,[2] and Project Gutenberg,[3] digitization has been actively organized by public institutions covering a wide range of documentation including historical archives, medical reposts, and government document repositories. However, poor Optical Character Recognition (OCR) accuracy is the primary issue that affects the presentation and analytics of the digitized texts (Alex & Burns, 2014; Lam-Adesina & Jones, 2006; Lopresti, 2009; Reynaert, 2014; Thompson et al., 2016). It is especially problematic for documents that require high reliability, such as medical information (Thompson, McNaught, & Ananiadou, 2015) and government records (Pereda & Taghva, 2011).

However, the scanned image files have different qualities, layouts, and font types which introduce errors that may be hard to avoid in a unified OCR workflow that aims for both efficiency and generality. Also, OCR engines that typically apply dictionary-based validation method has limited correction capability (Smith, 2007).

OCR post-processing is the procedure that aims to fix the residual errors in the OCR-generated text. The general practice of post-processing error correction on the OCR-generated text is agnostic to OCR engines and in the absent of the original scanned document image. An OCR post-processing model is thus able to be applied to digitized texts or a digitization pipeline with any OCR engine. OCR post-processing techniques has been broadly researched for manuscripts of different languages (Al Azawi, Ul Hasan, Liwicki, &

* Corresponding author.
E-mail addresses: jmei@cs.dal.ca (J. Mei), aminul@louisiana.edu (A. Islam), amohd@cs.dal.ca (A. Moh'd), yajing@cs.dal.ca (Y. Wu), eem@cs.dal.ca (E. Milios).
[1] https://books.google.com.
[2] https://www.biodiversitylibrary.org.
[3] https://www.gutenberg.org.

https://doi.org/10.1016/j.ipm.2018.06.001
Received 1 November 2017; Received in revised form 21 April 2018; Accepted 1 June 2018
0306-4573/ © 2018 Elsevier Ltd. All rights reserved.

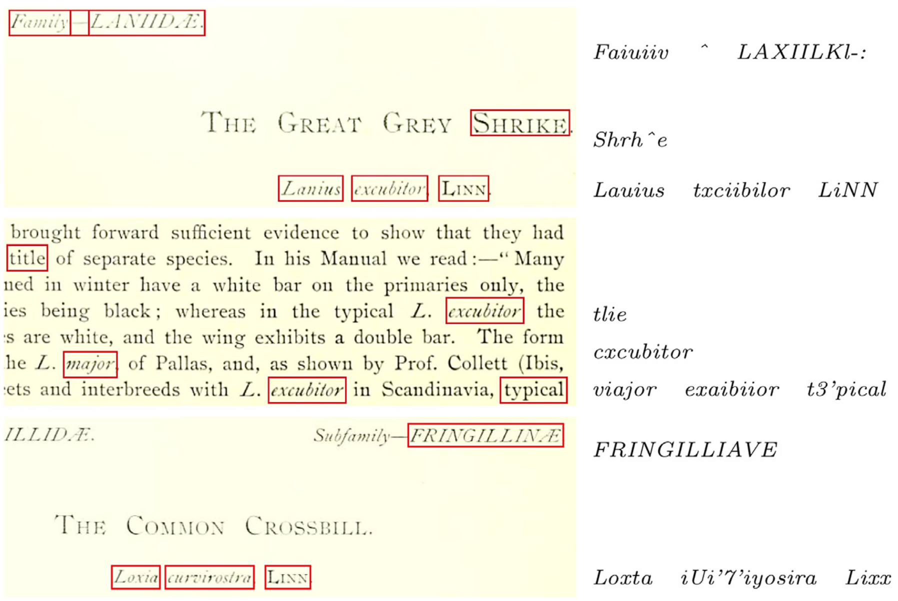

Family—LANIDÆ.
THE GREAT GREY SHRIKE.
Lanius excubitor, LINN.
brought forward sufficient evidence to show that they had
title of separate species. In his Manual we read:—" Many
ned in winter have a white bar on the primaries only, the
ies being black; whereas in the typical L. excubitor the
s are white, and the wing exhibits a double bar. The form
he L. major of Pallas, and, as shown by Prof. Collett (Ibis,
ets and interbreeds with L. excubitor in Scandinavia, typical
ILLIDÆ.
Subfamily—FRINGILLINÆ.
THE COMMON CROSSBILL.
Loxia curvirostra, LINN.
Faiuiiv ˆ LAXIILKl-:
Shrhˆe
Lauius txciibilor LiNN
tlie
cxcubitor
viajor exaibiior t3'pical
FRINGILLIAVE
Loxta iUi'7'iyosira Lixx

Fig. 1. Sample text image segments and corresponding errors in the OCR-generated text. The original text images corresponding to the OCR-generated errors are highlighted in red. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Breuel, 2014; Doush & Al-Trad, 2016; Reynaert, 2014).

Given only OCR-generated text as input, OCR error correction is arguably similar to correcting spelling errors. Conversely, OCR errors are inherently more volatile and complex than the misspellings generated by humans. First of all, misrecognized characters in OCR errors are not limited to letters, *e.g.*,⟨*LANIDÆ* → *LAXIILKl − :*⟩, which leads to more ambiguous error boundaries and hardly been handled correctly using the standard tokenization rules. Moreover, for handwriting or typing errors, the error patterns are usually being understood by the combination of Damerau–Levenshtein edit operations. However, a large portion of OCR errors violate such one-to-one character modification, *e.g.*,⟨ *viajor* → *major*⟩ and ⟨ *exaibiior* → *excubitor*⟩ as shown in Fig. 1. Most importantly, the generation of OCR errors results from the combined effects of various factors, including algorithmic defects (*e.g.*,segmentation, classification inaccuracy), limited hardware conditions (*e.g.*,poor scanning equipment), and complex content status (*e.g.*,mixture of text fonts, complicated page layout). It thus leads to volatile error patterns, *e.g.*,⟨ *excubitor* → *txciibilor, cxcubitor, exaibiior*⟩, or sophisticated errors that are orthographically far from their corrections, *e.g.*,⟨ *curvirostra* → *iUi' 7' iyosira*⟩. Hence the causal relation of errors generation is hard to be captured directly using a probabilistic model, especially with limited training instances.

In our OCR post-processing model, we design the entire correction pipeline to handle OCR-specific error patterns. Noisy text tokenization is known to have issues in word boundary detection (Kukich, 1992). Tokenization schemes using space and punctuation inevitably lead to merging (*e.g.*, *ofthe*) and splitting (*e.g.*, *j ust* or *typir,al*) errors. To deal with ambiguous word boundaries in noisy OCR text, we propose a two-stage tokenization scheme, which first utilizes a vocabulary to assert fuzzy boundary detection, and then adopts Peen Treebank conventions to normalize tokens. A comparison experiment shows that our adopted tokenization scheme significantly outperforms other methods on noisy OCR text.

Given the complex nature of OCR errors, we utilize rich linguistic features in lexical, semantics, and context to comprehensively support decision making for error detection and correction. These linguistic features are inferred from web-scale corpora, such as Google Web 1T corpus[4] and English Wikipedia[5] article titles. We use a recall-oriented classification to maximize the error identification probability and adopt ensemble regression to leverage these features for candidate ranking. The quantitative analysis of feature importance shows that *n*-gram context is important for error detection and a combined effort from different feature types is necessary for correction.

To increase the chance of finding corrections for volatile error patterns, we use two methods to explore candidates in Google Web 1T Corpus. We search candidates that are within three reverse Levenshtein distance to the error token from the unigrams of the Google Web 1T corpus. We also collect candidates using similar *n*-gram contexts in bigrams to 5-grams of the Google Web 1T corpus. While these two methods are capable of suggesting a large number of candidates, the computational cost for processing these candidates is overwhelming. We thus use feature-based ranking to prune the candidate set, which considerably reduces its size while maintaining the most appropriate candidates.

[4] https://catalog.ldc.upenn.edu/ldc2006t13.
[5] https://en.wikipedia.org/wiki.

To sum up, the contributions of our proposed OCR post-processing model are four-fold.

- We employ a tokenization scheme that can better deal with the boundary ambiguity to reduce the tokenization errors propagated into the downstream processes.
- For error detection, our model first reduces misidentified errors using a recall-oriented classifier, and then increases the detection accuracy by filtering the false predicted errors.
- Our model effectively locates correction within a very small candidate set by extensively exploring candidates in web-scale *n*-gram corpora and pruning them with linguistic features.
- We show that high quality suggestion can be achieved by leveraging diversified linguistic features using an ensemble regressor, which is robust to handle different scenarios including complex errors and false predicted errors.

The evaluation on post-processing an OCR-generated historical biology text demonstrates that our proposed correction model outperforms existing models in both its automatic mode (which selects the top suggestion as the correction) and its user-interactive mode (which presents a short list of suggestions to the user from which to select the correction). We conduct quantitative analysis to evaluate the error correction performance of the proposed post-processing pipeline. Finally, we discuss potential future directions for improving this model.

After pretrained by sufficient amount of data, our model is applicable for correcting documents with the same word distribution (*i.e.*,in the same domain as the training data) and error distribution (*i.e.*,generated from the same OCR engine). As our model achieves good performance and requires no additional human error in automatic correction, it is well-suited for large-scale digitization.

2. Related works

OCR post-processing models can be categorized as manual, semi-automatic, and automatic according to the degree of automation. Given the complexity of OCR errors, involving some degree of user interaction is a promising way to ensure the correction performance and is widely adopted in the mainstream post-processing applications, *e.g.*, Vobl, Gotscharek, Reffle, Ringlstetter, and Schulz (2014). Considering OCR post-processing uses only OCR-generated text, most techniques used for correcting OCR errors are derived from spelling correction methods.

Manual post-processing models rely fully on human efforts to validate and correct the errors in OCR-generated text. This type of models typically provides a Graphical User Interface (GUI) to a community with a large number of contributors working on post-processing a large OCR-generated archive (Causer, Tonra, & Wallace, 2012; Holley, 2009). To facilitate this user-centered workflow, a full-text search tool that can retrieve all occurrences of original images given a text query with an edit distance threshold is introduced by Mühlberger, Zelger, and Sagmeister (2014). Using this tool, contributors can directly search suspicious strings and make corrections to all the occurrences of an error. Manual models are useful to control the correction performance, especially for texts with high word error rate. However, their application scenarios are very limited given the substantial human effort required.

Studies on semi-automatic approach view the model processing as the initial step in the error correction pipeline and involve continuous human intervention afterwards. These models are designed to reduce the human effort of manually correcting errors. Integration of dictionaries and heuristics to correct as many OCR errors as possible before giving the text to human correctors is performed in Vobl et al. (2014). On that basis, the previous human selection is recorded to update the underlying Bayesian model for better correction performance is introduced by Taghva and Stofsky (2001). To better assist human interaction, GUI interfaces are usually provided to allow users to navigate the text and modify the detection and correction results (Taghva & Stofsky, 2001; Vobl et al., 2014).

Automatic models attempt to fix OCR text without human interaction. Some works focus on correcting some specific error types. A rule-based method that utilizes lexicon for correcting non-word errors is introduced by Reynaert (2011). Two commonly observed issues in historical OCR text – End-of-line hyphens and $\langle f \rightarrow s \rangle$ recognition errors – are studied in Alex, Grover, Klein, and Tobin (2012).

One class of automatic works ensembles multiple OCR output of the same input image and selects the best recognition for each word as the final output. Combining complementary recognition results from different OCR engines is claimed to lead to a better output (Lund & Ringger, 2009). The overall error rate is demonstrated to decrease with any additional OCR output regardless of its individual quality (Lund, Walker, & Ringger, 2011). A supervised prediction model that selects the best word recognitions among different OCR outputs is introduced by Lund, Douglas, and Ringger (2013). Combining OCR outputs with word lexical features and training a Conditional Random Fields model for word selection is further studied in Lund, Ringger, and Walker (2014). While such models have proved useful, they select words only among OCR-generated recognitions and are blind to other candidate words. Besides, they require the presence of the original image file, which reduce the generality of OCR post-processing.

Another type of automatic models leverage linguistic features in an supervised predictor for error detection and correction. This model setup is proved to be robust in different error correction applications (Bao, Kimelfeld, & Li, 2011; Gao, Li, Micol, Quirk, & Sun, 2010; Islam & Inkpen, 2011). A correction model that uses anagram hashing to handle lexical variation in a large text collection is introduced in Reynaert (2011). Given limited training corpora is available, statistics from external corpora brings great benefit. A model that utilize Google Web unigram corpus for detection and 5-gram corpus for correction is introduced by Bassil and Alwani (2012). Based on previous two models, a statistical-based learning model, which makes use of the character confusion frequencies and contextual features extracted from three million documents and employs a linear regressor for candidate ranking is proposed by Kissos and Dershowitz (2016). Correction candidates suggested by this model are not restricted to those found in OCR

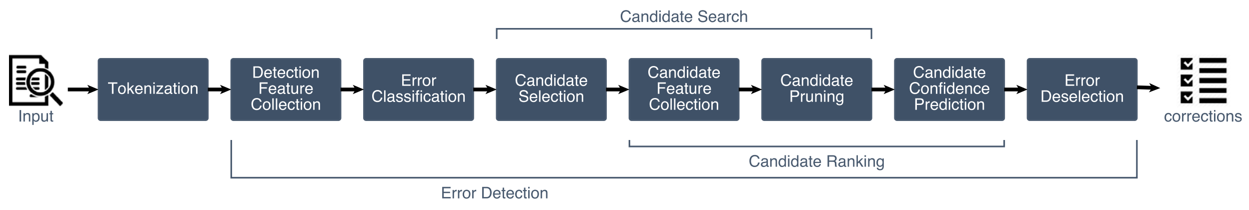


Fig. 2. The computational pipeline of the proposed OCR post-processing model. In Candidate Pruning, we reduce the size of the candidate set by feature-based ranking and pruning. In Error Deselection, given the ranked candidate list, we may re-identify a classified error to instead be correct if its top suggested candidate is the original word.

outputs. However, existing methods make use of solely n-gram frequencies without knowing the characteristics of OCR errors and are, thus, biased to select common words from the n-gram corpus.

3. Proposed model

The conventional error correction workflow contains three phases: *error detection* from text, *candidate correction generation* for the detected errors, and *candidate ranking* (Kukich, 1992). Our proposed model follows this paradigm and implements these phases in an interdependent process, as shown in Fig. 2.

3.1. Tokenization

In OCR-generated text, intra-word characters can be misrecognized as punctuation, which is hard to disambiguate with true punctuation and thus lead to higher token boundary ambiguities. Besides, tokenized units should also be consistent with the external corpus used in the downstream process. Without loss of generality, we accord with Google n-gram tokenization,[6] which is based on the Penn Treebank tokenization and includes additional rules to handle special tokens (*e.g.*,hyphenated words, urls).

To better handle these two objectives, we follow Fares, Oepen, and Zhang (2013) to separate tokenization into boundary detection and text normalization. We first segment text into subsequences by whitespace characters and then normalize each subsequence following Algorithm 1.

3.2. Detection feature collection

The error detection phase is responsible for identifying error tokens in the tokenized input text. For each token, we infer the correctness of a token from the following four types of features:

- *Word validity.*A correct token should be a valid word in the vocabulary. We use a binary value to indicate whether a token is included in the language vocabulary. This feature has been widely adopted to detect non-word errors. We should use vocabulary with high coverage of words in the language.
- *Character existence.* To capture the misrecognized letter being classified as punctuation, we use 33 binary values to represent the existence of punctuation characters.[7]
- *Exact context coherence.*A correct word should be appropriate in its context. This property is broadly adopted to language modeling and identify real-word errors. We propose to infer context coherence from contextual word n-grams statistics. We first search frequencies in the corpus for n-grams that consist of the pivot token and its surrounding tokens shown in Fig. 3. Let $\{g_1, g_2, \cdots, g_n\}$ be a set of frequency counts of exact contexts in an n-gram corpus, the context coherence of token w is estimated using the following formula:

$$\phi_{n\text{-gram}}(\mathrm{w}) = \frac{1}{Z_n}\sum_{i}^{n}\log(g_i + 1), \tag{1}$$

  where Z_n is a normalization factor that linearly rescale the feature values into range [0, 1] to avoid the adaption issue. Considering the long tail distribution of word frequencies, each n-gram frequency is mapped to the log space with add-one smoothing before rescaling. Existing n-gram contexts strongly infers the correctness of a word especially with a large context size n. This measure of the context coherence with log scale is sensitive to the existence, not to the frequencies, of the n-gram contexts. This measure is still able to distinguish the relative popularity of a word.
- *Approximate context coherence.*While context coherence with word n-gram has been shown to be an effective detection inference, an n-gram corpus always has limited coverage. It is especially problematic for distinguishing valid contextual n-grams that contain rare words. We therefore propose to relax one context word from the exact context, where a relaxed word can be substituted by

[6] https://catalog.ldc.upenn.edu/docs/LDC2006T13/readme.txt.

[7] Punctuation characters refer to non-alphabet and non-digit ASCII symbols in code range 32 to 126.

```
Input  : A whitespace segmented character sequence $\mathbf{s}$
         A language vocabulary $\mathcal{V}$
Output : A list of normalized tokens $\mathcal{S} = [\mathbf{s}_1, \ldots, \mathbf{s}_n]$, $s.t.$ concat($\mathcal{S}$) = $\mathbf{s}$

1  $\mathcal{S}^{\text{lhs}}, \mathbf{s}^*, \mathcal{S}^{\text{rhs}} \leftarrow$ truncate($\mathbf{s}$) ;        // truncate punctuation from both sides of $\mathbf{s}$ as tokens.
2  $\mathcal{S}^* \leftarrow$ dehyphenate($\mathbf{s}^*$) ;        // segment $\mathbf{s}^*$ at both sides of each hyphen character.
3  if $s_i^* \in \mathcal{S}^*$ $s.t.$ $s_i^* \notin \mathcal{V}$ then
4      $\mathcal{S}^* \leftarrow [\mathbf{s}^*]$
5  end
6  if $s_1^* \notin \mathcal{V}$ then
7      $\mathcal{S}^{\text{lhs}}, \mathcal{S}^* \leftarrow \phi$, [concat($\mathcal{S}^{\text{lhs}}, \mathbf{s}_1^*$), $\mathbf{s}_2^*, \ldots, \mathbf{s}_n^*$]
8  end
9  if $s_n^* \notin \mathcal{V}$ then
10     $\mathcal{S}^*, \mathcal{S}^{\text{rhs}} \leftarrow [\mathbf{s}_1^*, \ldots, \mathbf{s}_{n-1}^*$, concat($\mathbf{s}_n^*, \mathcal{S}^{\text{rhs}}$)], $\phi$
11 end
12 $\mathcal{S}^* \leftarrow$ split_contract($\mathcal{S}^*$) ;        // split PTB style contractions.
13 $\mathcal{S}^* \leftarrow$ merge_abbr($\mathcal{S}^*$) ;        // merge common abbreviations.
14 $\mathcal{S} \leftarrow [\mathcal{S}^{\text{lhs}}, \mathcal{S}^*, \mathcal{S}^{\text{rhs}}]$
```

Algorithm 1. OCR subsequence normalization.

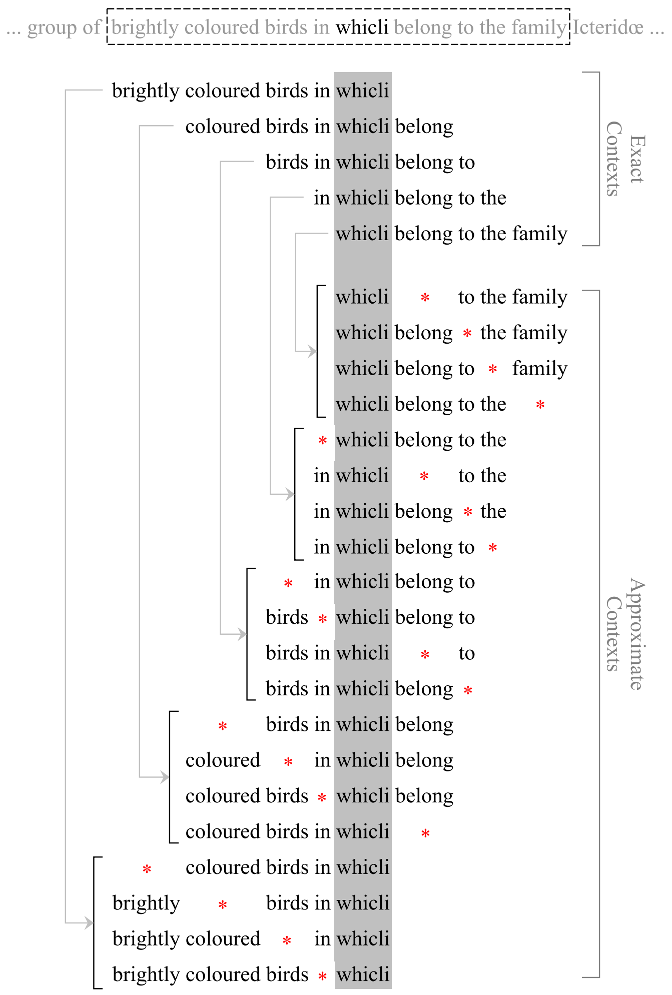

... group of brightly coloured birds in whicli belong to the family Icteridœ ...
brightly coloured birds in whicli
coloured birds in whicli belong
birds in whicli belong to
in whicli belong to the
whicli belong to the family
Exact Contexts
whicli * to the family
whicli belong * the family
whicli belong to * family
whicli belong to the *
* whicli belong to the
in whicli * to the
in whicli belong * the
in whicli belong to *
* in whicli belong to
birds * whicli belong to
birds in whicli * to
birds in whicli belong *
* birds in whicli belong
coloured * in whicli belong
coloured birds * whicli belong
coloured birds in whicli *
* coloured birds in whicli
brightly * birds in whicli
brightly coloured * in whicli
brightly coloured birds * whicli
Approximate Contexts

Fig. 3. 5-gram context collection example for token *whicli* in an OCR-generated text. Using a sliding window of size five, we are able to construct five exact 5-gram contexts. Substituting one context word with any valid word in the vocabulary, each exact 5-gram context is able to construct four approximate context patterns. In contextual candidate search and ranking, the same context collection step is performed after *whili* is substituted by a candidate word.

any valid word in the vocabulary (example in Fig. 3). The approximate context coherence is estimated using Eq. (1) with the frequencies of $n^2 - n$ approximate context.

3.3. Error classification

We leverage linguistic features in a classifier to predict its correctness. Given a tokenized text, the classifier should identifies tokens that likely to be OCR errors. Consider correct words will not be processed in the correction phase, the primary goal of the classifier is to reduce the number of false negative instances in error prediction. We therefore are biased to the recall of the error class instead of the overall accuracy. An obvious drawback of this prediction bias is to generate a large population of false position at the same time. We will solve this issue in the Error Deselection phase (Section 3.8).

To achieve high recall, classifiers inevitably generate more false positive predictions. To further select classifier that can effectively distinguish the error class, we conduct model selection using the evaluation metric based on the Bookmaker Informedness (BI). The Bookmark Informedness is given by:

$$\text{BI}=\frac{\text{TP}}{\text{TP + FN}} - \frac{\text{FP}}{\text{FP + TN}}, \tag{2}$$

where TP, FP, TN, FN are the number of true positive, false positive, true negative, and false negative predictions, respectively. We use the transformation of BI with two penalty terms on FN and FP instances and assign weights to differentiate their importance in classifier evaluation:

$$J = 1 - \alpha\,\frac{\mathrm{FN}}{\mathrm{TP} + \mathrm{FN}} - (1 - \alpha)\frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}. \tag{3}$$

Here α can also be treated as a sensitive coefficient for deciding the trade-off between the population of FN and FP instances. As we are more concern about FN, α is set to be greater than 0.5. Experimentally, we adopt $\alpha = 0.65$.

3.4. Candidate selection

To include the correction of complex error patterns in the candidate set, we adopt two different methods for selection.

- *Reverse-Levenshtein distance search.*We adopt Reverse-Levenshtein search to explore words in the vocabulary that are within a limited Levenshtein distance to the error word.
- *Contextual search.*Besides candidates that are orthographically similar to the error, we also consider candidates that are coherent to the context. We construct exact/relax contexts after the error token is substituted by any valid word in the vocabulary, and collect the substituted word as a candidate if any context exists in the *n*-gram corpus.

Unlike other models that restrict the size of the candidate sets, we consider a broad range of candidates in this step by using a loose distance threshold and searthing in a large *n*-gram corpus with different context length. We refine the candidate set in the subsequent procedures.

3.5. Candidate feature collection

To broadly represent the word characteristics and their relation with the error, we profile each selected candidate by a vector of linguistic statistics in the following six feature types, where a full list of 39 feature values are shown in Table 1.

- *Candidate popularity*. A candidate that is frequently used in a language is potentially more likely to be a correction. We characterize the popularity of a candidate using the relative unigram frequency ratio among all candidates that are suggested for this error token. Let $\mathscr{C} = \{c_1, c_2, \cdots, c_n\}$ be a set of unigram frequencies of suggested candidates given error token w, the popularity of candidate w_i with frequency $c_i \in \mathscr{C}$ is given by:

Table 1
List of 39 applied candidate features in six feature types. A feature type may have different measures and computational settings, where *size* indicates a set of different word/character *n*-gram size.

Feature type (Num. of features)	Settings
Candidate popularity (1)	
Lexicon existence[a] (3)	*lexicon* = {*wiki, term, bio*}
Edit distance (4)	
LCS distance	
Levenshtein distance	
Damerau–Levenshtein distance	
Optimal String Alignment	
Character *n*-gram distance (21)	
n-gram Jaccard coefficient	*size* = {1, 2, 3, 4, 5}
q-grams distance	*size* = {2, 3, 4, 5}
n-grams distance[b]	*size* = {2, 3, 4, 5}
	type = {*bin, pos, comp*}
Lexical similarity (2)	
String Similarity	
Jaro–Winkler distance	
Context coherence (8)	
Exact context	*size* = {2, 3, 4, 5}
Approximate context	*size* = {2, 3, 4, 5}

[a] Three lexicon existence settings use different external resources: Wikipedia title list (*wiki*, domain-specific glossaries (*gls*)), and biodiversity terminology list (*bio*) as we evaluated on an OCR-generated biodiversity book (see Section 4.1).

[b] There are three overlapping measurements proposed for *n*-gram distance, namely binary (*bin*), positional (*pos*), and comprehensive (*comp*). Combining each overlapping measurement with different character *n*-gram size, there are in total 12 *q*-gram feature values.

Table 2
The Levenshtein distance distribution of the errors in the evaluation dataset. There is one sample error of each distance being selected from the proposed dataset to showcase the error pattern.

Edit distance	Error statistics		Sample OCR error	
	Num.	Percent [%]	Correction	Error
1	889	30.58	galbula	ga/bula
2	1376	47.35	yellowish	j^ellowish
3	307	10.56	bents	Ijcnts
4	148	5.09	my	ni}’
5	70	2.41	Lanius	Lioiiits
6	51	1.75	minor	)iii > iof
7	28	0.96	garrulus	f;ay > ///us
8	16	0.55	curvirostra	iUi’7’iyosira
9	5	0.17	Nucifraga	Aiiii/rut^d
≥ 10	16	0.55	pomeranus	poiiui-iVtiis
Total	2906	100.00		

$$\phi_{\mathrm{cp}}(\mathrm{w}_i) = \frac{\log(c_i + 1)}{\log(\max(\mathscr{C}) + 1)}. \tag{4}$$

- *Lexicon existence*. To identify whether a token is included by a specific semantic group, we use domain specific lexicons of different coverage, *i.e.*,Wikipedia article titles, domain-specific glossaries, and biodiversity terminologies, to identify whether a candidate relates to the topic and to avoid the situation in which rare terminologies are underestimated by other features.
- *Edit distance*. Edit distance measures, which count the number of edit operations required to transform between two strings, are fundamental techniques in quantifying the lexical difference. Different edit distance measures allow decomposition string transformation using different set of edit operations or with additional restrictions. To quantify different aspects of lexical difference between an error token and a suggested correction candidate, we make a separate feature for each edit distance measure, including LCS distance, Levenshtein distance, Damerau–Levenshtein distance, and optimal string alignment (Boytsov, 2011).
- *Character n-gram distance*. As many OCR errors involve incorrect image segmentation that affect multiple characters, *e.g.*,⟨ *ni}'* → *my*⟩, we adopt the distance measure based on character *n*-grams, including *n*-gram based Jaccard coefficient, *q*-gram distance (Navarro, 2001), and *n*-gram distance (Kondrak, 2005).
- *Lexical similarity*. We also apply other lexical relatedness measures to enrich the feature set. Jaro–Winkler (Cohen, Ravikumar, & Fienberg, 2003) is a metric that suitable for measuring phrases – such as person or place names – and favorable to string pairs with common prefix. String similarity measure (Islam & Inkpen, 2009) that takes the weighted sum of different LCS variations are also used.
- *Exact/approximated context coherence*. We adopt exact and approximated context coherence feature as described previously, where the error token is substituted by the suggested candidate.

3.6. Candidate pruning

With feature values being evaluated, candidates can be ranked by the preference of different linguistics aspects. To further select the most appropriate candidates from potentially million-scale candidate set of each error, we union the top selected candidates ranked by different candidate features.

3.7. Candidate confidence prediction

We adopt a regressor to leverage the candidate features for ranking in the pruned candidates set. To train this regressor, we follow our previous described steps to detect and search candidates for the potential errors in the training text. We then assign binary label to each candidate, where only the correct candidates are labeled as 1, and the union of all candidate set is used for training. Since a correct word can be misclassified as an error, we label its original word to 1 if it is included in the suggested candidate set.

3.8. Error deselection

Given recall oriented error classification, there are more FP predictions being generated, which will increase human labor in an interactive correction setup. To revise this detection bias, we rejudge the suggested corrections. Given candidate corrections suggested by the model, an error word is relabeled as correct if its top ranked candidate is the original word.

Table 3
Model comparison on error detection and correction. Error detection is evaluated by precision (Prec.), recall (Rec.), and F1 measure. Error correction is evaluated by different precision measures representing the automatic correction performance (*P*@1) and user-interactive correction performance (*P*@3, *P*@5, and *P*@10), where *P*@*n* indicates the correction precision given the top *n* model suggested candidate for each error.

Error correction model	Measures [%]						
	Error detection			Error correction			
	Prec.	Rec.	F1	P@1	P@3	P@5	P@10
Aspell [bad-spellers]	71.20	87.81	79.34	23.64	23.64	23.64	23.64
Aspell [normal]	71.20	87.81	79.34	24.03	24.03	24.03	24.03
Aspell [fast|ultra]	71.20	87.81	79.34	25.83	25.83	25.83	25.83
Noisy Channel (Segaran & Hammerbacher, 2009)	51.20	80.81	66.81	40.80	45.81	55.81	55.81
Kissos and Dershowitz (2016)	**71.75**	83.51	77.22	51.62	63.51	61.62	63.51
proposed approach	70.56	**91.47**	**80.17**	**52.14**	**66.88**	**68.94**	**71.95**

4. Evaluation

4.1. Evaluation dataset

It is hard to directly compare a new OCR post-processing model with the existing models because the benchmark testing datasets mentioned in the literature are not publicly available. We thus created a dataset for OCR post-processing evaluation and made it publicly available.[8] The OCR text in the dataset was generated from the content of the book titled “Birds of Great Britain and Ireland (Volume II, 1907, 274 pages)”[9] and made publicly available by the Biodiversity Heritage Library (BHL) for Europe using Tesseract 3.0.2. We manually correct the text to generate the ground truth and list all the OCR errors with their corrections. The Levenshtein distance distribution of the error words, shown in Table 2, provides an evidence that a large portion of the OCR errors have complex error patterns.

4.2. Experimental setup

We use the OCR text containing the first 80% errors in the ground truth dataset for training and the remaining 20% for testing. For all the experiments conducted in this paper, we use English Wikipedia titles as the vocabulary and Google Web 1T dataset as the *n*-gram corpus.

Considering the tokenization errors, tokens generated from the training text are labeled according to whether they overlap with any ground truth error. All labeled tokens are used to train an Extremely Randomized Tree[10] classifier, where hyperparameters are optimized using random search (Bergstra & Bengio, 2012) and each hyperparameter setting is evaluated by Eq. (3) with 10-fold cross-validated predictions. As the supervised candidate ranker relies on the errors predicted by the classifier, the domain adaptation issue arises if we train the ranker using the ground truth error while testing with the predicted errors. We thus use the learned classifier to predict errors in the training tokens and use the predicted errors to train a Gradient Boosting regressor for candidate ranking.

4.3. Performance evaluation

We evaluate the error detection performance using precision (Prec.), recall (Rec.), and F1 measure. We also evaluate the correction performance using the correction precision given the top *n* candidate suggestions (P@*n*). In Table 3, we compare the proposed model against the following three baseline models that use different correction approaches:

- *Aspell* is an open-source spell checker preinstalled in Linux distributions, which suggests candidates that are phonetically similar and within two Damerau–Levenshtein distance to the error. It has different modes – namely ultra, fast, normal, bad-spellers – in seeking different balance points between running speed and candidate coverage. This experiment uses Aspell version 0.60.7 with updated English dictionaries from SCOWL[11] Version 2017.01.22.

[8] https://github.com/jmei91/MiBio-OCR-dataset.
[9] https://www.biodiversitylibrary.org/item/35947#page/19/mode/1up.
[10] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html.
[11] http://wordlist.aspell.net/.

Table 4
Sample of processed error tokens with the top three candidates suggested by each model. True Positive (TP) errors and their corresponding corrections are annotated as ⟨error → correction⟩. A correction model may split the error word into multiple tokens and make correction for each one. A token has no candidates suggestion if it is not been detected as an error.

Model		Detection status								
		TP							FP	
		⟨*iDVCiisis* → *arvensis*⟩	⟨*l)ut* → *but*⟩			⟨*j–ellowish* → *yellowish*⟩			*syllabled*	*gradational*
OCRrect	Detected span	*iDVCiisis*		*l)ut*		*j*	*-*	*ellowish*	*syllabled*	*gradational*
	Suggestion	*arvensis*		*but*		*to*	*-*	*yellowish*	*syllabled*	*gradational*
		is		*out*		*of*	*,*	*Yellowish*	*syllable*	*free*
		dists		*But*		*in*	*:*	*mellowish*	*described*	*early*
Aspell	Detected span	*iDVCiisis*	*l*	*)*	*ut*	*j*	*-*	*ellowish*	*syllabled*	*gradational*
	Suggestion	*advises*			*UT*			*yellowish*	*syllables*	*gradations*
		advisers			*IT*			*lowish*	*syllable*	*gradation*
		advice's			*It*			*elfish*	*syllable's*	*gradation's*
Noisy Channel	Detected span	*iDVCiisis*	*l*	*)*	*ut*	*j*	*-*	*ellowish*	*syllabled*	*gradational*
	Suggestion				*ut*	*j*	*-*	*yellowish*	*syllabled*	*gradational*
					at	*b*	*y*	*mellowish*	*syllables*	*graditional*
					it	*o*	*r*	*slowish*	*syllable*	*tradational*

- *Noisy channel* is a widely used probabilistic correction model, where suggestions are based on the probability of edit operations between candidates and errors. We adopt the implementation and pretrained dataset given by Segaran and Hammerbacher (2009).
- Kissos and Dershowitz (2016) proposed a statistical correction model, which makes use of the character confusion frequencies and contextual features extracted from three million documents and employs a linear regressor for candidate ranking.

In detection, the results show that the proposed approach significantly outperforms all the other models in recall and achieves a competitive precision. In correction, the proposed approach outperforms baseline methods with the top 1, 3, 5, and 10 suggested candidates.

4.4. Correction case study

To better illustrate the difference between approaches, we show some typical errors with candidate corrections suggested by different models in Table 4. Using a broad candidate selection and robust candidate ranking with rich linguistic features, our model is able to suggest candidates that are neither limited to lexicographic nor phonetic similarity. This is helpful for solving complex OCR errors such as ⟨*iDVCiisis* → *arvensis*⟩. For FP predictions produced in the classification phases, our model is capable of revising their detected labels given the correct top suggestion. Moreover, while our model tends to suggest a diversified candidate corrections, it correctly ranks the most appropriate candidates for errors. We observe that many uncorrected errors involve boundary detection issues in tokenization, our model better handles the boundary ambiguities in the OCR-generated text.

5. Discussion

In the proposed model, we applied techniques that are different from other existing models in the literature. We discuss the implications of these choices and conduct quantitative comparison against baseline methods.

5.1. Tokenization performance

Although error correction techniques usually assume that the boundaries of error words are correctly identified, noisy text tokenization still remains an unsolved challenge, and tokenization error inevitably propagates into downstream analysis as in examples shown in Table 4. We thus quantitatively investigate the performance of different tokenization schemes on OCR texts with respect to the ground-truth tokens in terms of precision, recall, F_1 score, and tokenization error rate (Err.), where Err. is defined as follows:

$$\text{Err.} = \frac{\text{FP} + \text{FN}}{\text{TP}+\text{FP}+\text{FN}}. \tag{5}$$

Table 5
Performance of different tokenization methods on the proposed dataset.

Tokenization method	Measure [%]			
	Prec.	Rec.	F_1	Err.
Whitespace convention[a]	85.50	73.14	78.84	34.93
Penn Treebank convention[a]	94.33	93.94	94.13	11.08
WASTE[b] (Jurish, 2013)	95.18	93.14	94.15	11.05
Elephant (Evang et al., 2013)	95.17	93.18	94.16	11.03
Proposed method	**99.04**	**98.98**	**99.01**	**1.96**

[a] The conventions are implemented in the Stanford CoreNLP toolkit (Manning et al., 2014).
[b] WASTE is pretrained using the British National Corpus.[12]

Table 6
Performance of different perdition models for detecting errors on the tokenized testing data. The models are implemented in the scikit-learn toolkit (Pedregosa et al., 2011).

Classification model	Prediction type			Measure	
	TP	FN	FP	Rec.	Eq. (3)
Logistic Regression	598	94	1743	0.8642	0.8896
Decision Tree	599	93	1574	0.8656	0.8927
Random Forest	627	65	2140	0.9061	0.8972
Extra Tree	633	59	2373	0.9147	**0.8983**
Isolation Forest	490	202	5305	0.7081	0.7430
One-Class SVM	643	49	8826	0.9292	0.7819

The results are shown in Table 5. While whitespace convention is used by the majority of the error detection and correction techniques (Kissos & Dershowitz, 2016; Kukich, 1992; Lund et al., 2011), its performance on OCR text is hardly acceptable. The Penn Treebank convention achieves relatively better performance since it can heuristically disambiguate English punctuation. In comparison with the two unsupervised models (i.e., whitespace convention and the Penn Treebank convention), two supervised tokenization models, WASTE (Jurish, 2013) and Elephant (Evang, Basile, Chrupała, & Bos, 2013), generate less tokens and receive higher precision. Presumably these models have learned intricate rules in identifying the real punctuation, and thus some fragmented words can be restored. However, they achieve limited improvement in tokenization performance. Given a significant better performance achieved by the proposed tokenization scheme, we posit that explicitly defined rules are well-suited to tokenize noisy text with intricately entangled error patterns.

5.2. Detection models comparison

We evaluate different classification models for error prediction on testing data with tokens generated from the proposed tokenization scheme. We train different classifiers using the setup described in the evaluation. The result in Table 6 shows that Extremely Randomized Trees (Extra Tree) outperforms other classification models when $\alpha = 0.65$. We observe from further experiments that different classifiers are preferred in different trade-offs (*i.e.*,α) between FN and FP. We also make comparison with outliers detection models including Isolation Forest[13] and One-class SVM.[14] However, given the detection features these methods fail to provide competitive results.

5.3. Selected candidate sets

As mentioned previously, candidate pruning aims to reduce the size of the candidate set while reserving the correction. To evaluate the effectiveness of candidate pruning, we use different thresholds for top selecting candidates ranked by features. Table 7 shows that over 80 percent of the corrections can be found from the top one candidate suggested by each feature, where more than 63 percent of the TP errors can be located. Compare with the original candidate sets (millions in size), the computation cost with the pruned candidate sets for the subsequent steps is significantly reduced.

[12] http://www.natcorp.ox.ac.uk/.
[13] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html.
[14] http://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html.

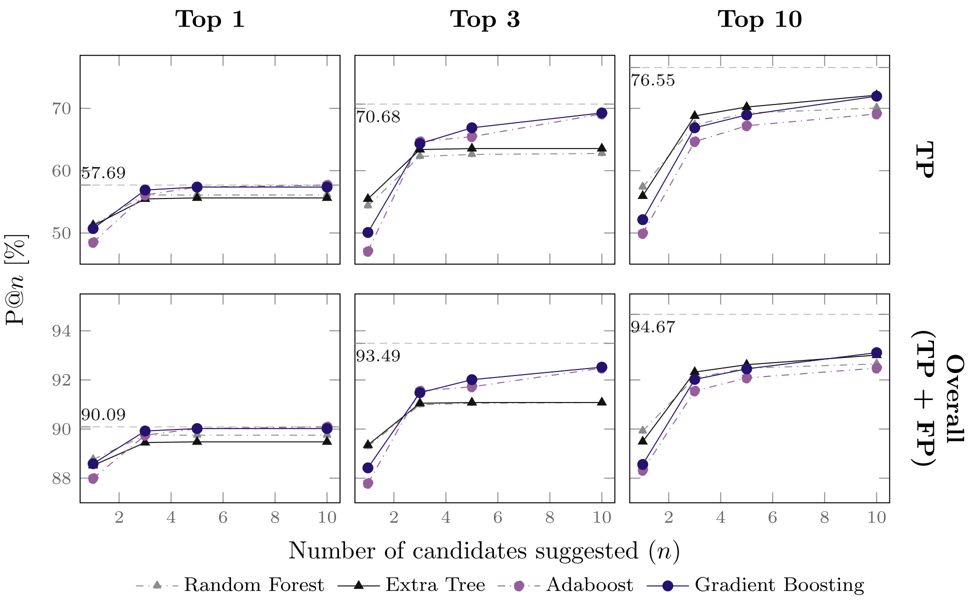


Fig. 4. Performance of four ensemble models for error correction. The top title of each plot (*i.e.*,Top 1, Top 3, and Top 10) indicates the type of the pruned candidate sets that are used for training. The right title of each plot (*i.e.*,TP and Overall) indicates the type of the error subset used for testing. The dash line on each plot represents the correction upperbound given by the corresponding pruned set in Table 7.

Table 7
The size and the correction coverage of candidates with different pruning thresholds.

Pruned testing set	Candidates		Coverage [%]	
	Avg.	Std.	TP	Overall
Top 1	8.22	1.85	57.69	90.09
Top 3	25.22	4.47	70.68	93.49
Top 5	41.33	6.83	74.01	93.95
Top 10	77.94	12.85	76.55	94.67
Top 100	5711.66	12673.54	80.51	96.17

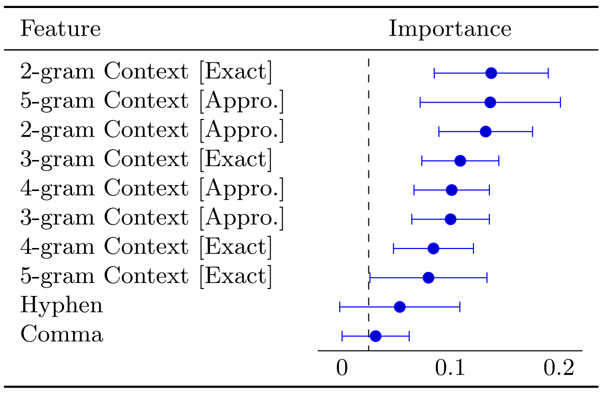
5.4. *Ranking models comparison*

We evaluate different regression models for candidate ranking using the proposed approach, where each model is trained with different pruned candidate sets. Among trained regressors, ensemble models are shown to be well-suited for this task. We compare the performance of four ensemble regressor tested on only TP and all the errors (TP + FP) in Fig. 4, where different hyperparameter settings are validated by P@1 of TP errors. With precision upperbounds given by the coverage of the training data, all four models can achieve high precision that are closed to the upperbound, where Gradient Boosting consistently outperforms other models in scenarios where $n > 1$.

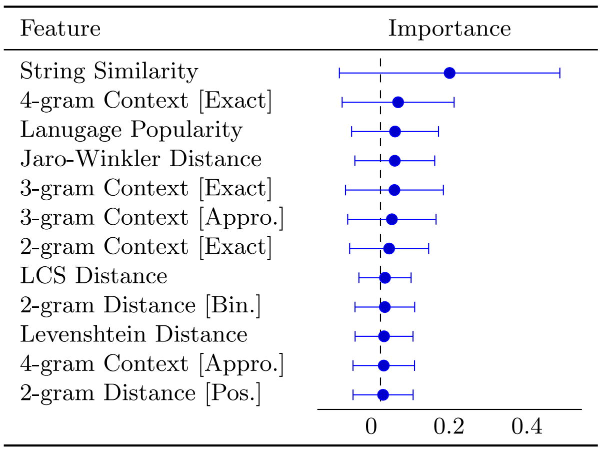
5.5. *Feature importance*

To get a better intuition of feature contribution on error detection, we explore the feature importances in the Extremely Randomized Trees model trained with the best validated setting. We list the most important features of the ensembled trees along with their inter-trees variability in Fig. 5a. Interestingly all context coherence statistics are the most influential features in identifying errors. Hyphen and comma are also strong indicators for some errors, such as ⟨*ni*}_ → *my*⟩ and ⟨au}*'* → *any*⟩.

While dictionary look-up is proven to be successful in non-word error detection, word validity feature plays a minor role in the proposed approach. We also anecdotally note that word validity is the most important feature in accuracy oriented settings. Presumably this feature could accurately distinguish different classes though tend to give false labels on marginal cases and as a result lower the recall.



(a) Most important detection features ($n = 42, \mu = 0.0238$).



(b) Most important correction features ($n = 39, \mu = 0.0256$).

Fig. 5. List of features with importance values are above the average. The dash line indicates the average importance of all applied features.

With the pruned top 10 candidate sets, we train the best validated setting of Gradient Boosting and list the most important tree-based features shown in Fig. 5b. String similarity feature is the most important feature in candidates suggestion, as it captures the common subsequences between the error and the candidates. Unlike the detection model, the ranking model relies on a diversified types of features and the importance of all the features witnesses a high variance among ensembled trees. These characteristics indicate that different linguistic features are required to achieve a robust ranking model for correction.

6. Conclusion and future work

We introduce a statistical learning model for correcting OCR-generated errors. By using an interdependent correction pipeline that continuously refines the error detection and list of suggested candidate corrections, the proposed approach is able to make use of web-scale corpora and integrate rich lexical, semantic and context features. Comparative experiments with baseline models show that our model can achieve significantly higher detection recall and suggest high quality candidates for correction. Quantitative analysis of the results of each computational step further supports that our proposed model is well-suited for handling volatile and complex OCR error patterns.

In our model, linguistic features play an important role in error identification, candidate set reduction, and candidate ranking. Exploring additional features remains an avenue for future work. Additionally, given that our model involves hyperparameters in different processing steps, it would be interesting to find a parameter setup that can achieve optimal performance.

Acknowledgment

This research was funded by the Social Sciences and Humanities Research Council of Canada (SSHRC) via RGPDD 451330, RGPIN 130082, and RGPIN 06183.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ipm.2018.06.001.

References

Al Azawi, M., Ul Hasan, A., Liwicki, M., & Breuel, T. M. (2014). Character-level alignment using WFST and LSTM for post-processing in multi-script recognition systems - a comparative study. In A. Campilho, & M. Kamel (Eds.). *Proceedings of the 11th international conference on image analysis and recognition (ICIAR): Part I* (pp. 379–386). Cham: Springer International Publishing.

Alex, B., & Burns, J. (2014). *Estimating and rating the quality of optically character recognised text. Proceedings of the first international conference on digital access to textual cultural heritage (DATeCH)*. Madrid, Spain: ACM97–102. http://dx.doi.org/10.1145/2595188.2595214.

Alex, B., Grover, C., Klein, E., & Tobin, R. (2012). Digitised historical text: Does it have to be mediOCRe? In J. Jancsary (Ed.). *Proceedings of the 2012 conference on natural language processing (KONVENS) (LThist Workshop)* (pp. 401–409). ÖGAI.

Bao, Z., Kimelfeld, B., & Li, Y. (2011). *A graph approach to spelling correction in domain-centric search. Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies (HLT)1. Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies (HLT)* Portland, Oregon: ACL905–914.

Bassil, Y., & Alwani, M. (2012). Context-sensitive spelling correction using Google Web 1T 5-gram information. *Computer and Information Science, 5*(3), http://dx.doi.org/10.5539/cis.v5n3p37.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research (JMLR), 13*(1), 281–305.

Boytsov, L. (2011). Indexing methods for approximate dictionary searching: Comparative analysis. *Journal of Experimental Algorithmics (JEA), 16*. http://dx.doi.org/10.1145/1963190.1963191 1.1:1.1–1.1:1.91

Causer, T., Tonra, J., & Wallace, V. (2012). Transcription maximized; expense minimized? Crowdsourcing and editing the collected works of Jeremy Bentham. *Literary and Linguistic Computing, 27*(2), 119–137. http://dx.doi.org/10.1093/llc/fqs004.

Cohen, W., Ravikumar, P., & Fienberg, S. (2003). *A comparison of string metrics for matching names and records. KDD workshop on data cleaning and object consolidation3. KDD workshop on data cleaning and object consolidation* 73–78.

Doush, I. A., & Al-Trad, A. M. (2016). Improving post-processing optical character recognition documents with Arabic language using spelling error detection and correction. *International Journal of Reasoning-based Intelligent Systems (IJRIS), 8*(3/4), http://dx.doi.org/10.1504/IJRIS.2016.10003960.

Evang, K., Basile, V., Chrupała, G., & Bos, J. (2013). *Elephant: Sequence labeling for word and sentence segmentation. Proceedings of the 2013 conference on empirical methods in natural language processing (EMNLP)*. Seattle, Washington, USA: ACL1422–1426.

Fares, M., Oepen, S., & Zhang, Y. (2013). Machine learning for high-quality tokenization replicating variable tokenization schemes. In A. Gelbukh (Ed.). *Proceedings of the 14th international conference on computational linguistics and intelligent text processing (CICLing): Part 1* (pp. 231–244). Springer Berlin Heidelberg.

Gao, J., Li, X., Micol, D., Quirk, C., & Sun, X. (2010). *A large scale ranker-based system for search query spelling correction. Proceedings of the 23rd international conference on computational linguistics (COLING)*. Beijing, China: ACL358–366.

Holley, R. (2009). *Many hands make light work : Public collaborative OCR text correction in Australian historic newspapers.* National Library of Australia Staff Papers1–28 March

Islam, A., & Inkpen, D. (2009). *Real-word spelling correction using Google Web 1T n-gram with Backoff. Proceedings of the 2009 international conference on natural language processing and knowledge engineering (NLPKE)*. IEEE1–8. http://dx.doi.org/10.1109/NLPKE.2009.5313823.

Islam, A., & Inkpen, D. (2011). Correcting different types of errors in texts. In C. Butz, & P. Lingras (Eds.). *Proceedings of the 24th Canadian conference on artificial intelligence (CAI)* (pp. 192–203). Springer Berlin Heidelberg.

Jurish, B. (2013). Word and sentence tokenization with hidden Markov models. *Journal for Language Technology and Computational Linguistics (JLCL), 28*(2), 61–83.

Kissos, I., & Dershowitz, N. (2016). *OCR error correction using character correction and feature-based word classification. Proceedings of the 12th IAPR workshop on document analysis systems (DAS)*. IEEE198–203. http://dx.doi.org/10.1109/DAS.2016.44.

Kondrak, G. (2005). *N-gram similarity and distance. Proceedings of the 12th international conference on string processing and information retrieval (SPIRE)*. Buenos Aires, Argentina: Springer-Verlag115–126. http://dx.doi.org/10.1007/11575832_13.

Kukich, K. (1992). Techniques for automatically correcting words in text. *Journal ACM Computing Surveys (CSUR), 24*(4), 377–439. http://dx.doi.org/10.1145/146370.146380.

Lam-Adesina, A. M., & Jones, G. J. (2006). Examining and improving the effectiveness of relevance feedback for retrieval of scanned text documents. *Information Processing & Management (IPM), 42*(3), 633–649. http://dx.doi.org/10.1016/j.ipm.2005.06.006.

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. http://dx.doi.org/10.1007/s10032-009-0094-8.

Lund, W. B., Douglas, J. K, & Ringger, E. K. (2013). *Combining multiple thresholding binarization values to improve OCR output. Proc. SPIE 8658, document recognition and retrieval XX, 86580R*http://dx.doi.org/10.1117/12.2006228.

Lund, W. B., & Ringger, E. K. (2009). *Improving optical character recognition through efficient multiple system alignment. Proceedings of the 9th ACM/IEEE-CS joint conference on digital libraries (JCDL)*. Austin, TX, USA: ACM231–240. http://dx.doi.org/10.1145/1555400.1555437.

Lund, W. B., Ringger, E. K., & Walker, D. D. (2014). *How well does multiple OCR error correction generalize? Proc. SPIE 9021, document recognition and retrieval XXI, 90210A*http://dx.doi.org/10.1117/12.2042502.

Lund, W. B., Walker, D. D., & Ringger, E. K. (2011). *Progressive alignment and discriminative error correction for multiple OCR engines. Proceedings of the 2011 international conference on document analysis and recognition (ICDAR)*764–768. http://dx.doi.org/10.1109/ICDAR.2011.303.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., & McClosky, D. (2014). *The stanford CoreNLP natural language processing toolkit. Proceedings of 52nd annual meeting of the association for computational linguistics (ACL): system demonstrations*. Baltimore, Maryland: ACL55–60.

Mühlberger, G., Zelger, J., & Sagmeister, D. (2014). *User-driven correction of OCR errors: Combining crowdsourcing and information retrieval technology. Proceedings of the first international conference on digital access to textual cultural heritage (DATeCH)*. Madrid, Spain: ACM53–56. http://dx.doi.org/10.1145/2595188.2595212.

Navarro, G. (2001). A guided tour to approximate string matching. *ACM Computing Surveys (CSUR), 33*(1), 31–88. http://dx.doi.org/10.1145/375360.375365.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: machine learning in python. *Journal of Machine Learning Research (JMLR), 12*(Oct), 2825–2830.

Pereda, R., & Taghva, K. (2011). *Fuzzy information extraction on OCR text. Proceedings of the 2011 eighth international conference on information technology: New generations (ITNG)*. IEEE543–546. http://dx.doi.org/10.1109/ITNG.2011.99.

Reynaert, M. (2014). *On OCR ground truths and OCR post-correction gold standards, tools and formats. Proceedings of the first international conference on digital access to textual cultural heritage (DATeCH)*. Madrid, Spain: ACM159–166. http://dx.doi.org/10.1145/2595188.2595216.

Reynaert, M. W. C. (2011). Character confusion versus focus word-based correction of spelling and OCR variants in corpora. *International Journal on Document Analysis and Recognition (IJDAR), 14*(2), 173–187. http://dx.doi.org/10.1007/s10032-010-0133-5.

Segaran, T., & Hammerbacher, J. (2009). *Beautiful data: The stories behind elegant data solutions. Theory in practice*O'Reilly Media.

Smith, R. (2007). *An overview of the Tesseract OCR engine. Proceedings of the ninth international conference on document analysis and recognition (ICDAR)2. Proceedings of the ninth international conference on document analysis and recognition (ICDAR)* IEEE629–633. http://dx.doi.org/10.1109/ICDAR.2007.4376991.

Taghva, K., & Stofsky, E. (2001). OCRSpell: An interactive spelling correction system for OCR errors in text. *International Journal on Document Analysis and Recognition (IJDAR), 3*(3), 125–137. http://dx.doi.org/10.1007/PL00013558.

Thompson, P., Batista-Navarro, R. T., Kontonatsios, G., Carter, J., Toon, E., McNaught, J., et al. (2016). Text mining the history of medicine. *PLOS ONE, 11*(1), 1–33. http://dx.doi.org/10.1371/journal.pone.0144717.

Thompson, P., McNaught, J., & Ananiadou, S. (2015). *Customised OCR correction for historical medical text. Proceedings of the 2015 digital heritage international congress1. Proceedings of the 2015 digital heritage international congress* IEEE35–42. http://dx.doi.org/10.1109/DigitalHeritage.2015.7413829.

Vobl, T., Gotscharek, A., Reffle, U., Ringlstetter, C., & Schulz, K. U. (2014). *PoCoTo - an open source system for efficient interactive postcorrection of OCRed historical texts. Proceedings of the first international conference on digital access to textual cultural heritage (DATeCH)*. Madrid, Spain: ACM57–61. http://dx.doi.org/10.1145/2595188.2595197.