Post-Processing OCR Text using Web-Scale Corpora

Jie Mei[†], Aminul Islam[‡], Abidalrahman Moh'd[†], Yajing Wu[†], Evangelos Milios[†]

[†]Faculty of Computer Science, Dalhousie University
{jmei,amohd,yajing,eem}@cs.dal.ca
[‡]School of Computing and Informatics, University of Louisiana at Lafayette
aminul@louisiana.edu

ABSTRACT

We introduce a (semi-)automatic OCR post-processing system that utilizes web-scale linguistic corpora in providing high-quality correction. This paper is a comprehensive system overview with the focus on the computational procedures, applied linguistic analysis, and processing optimization.

CCS CONCEPTS

• **Applied computing** → **Document analysis**; • **Information systems** → *Data analytics*; • **Computing methodologies** → *Natural language processing*;

KEYWORDS

OCR Error Correction; OCR Post-Processing; Statistical Learning

1 INTRODUCTION

There are massive amounts of data – including magazines, books, and scientific articles – stored and transmitted in the digital image formats, such as Portable Document Format (PDF) or Joint Picture Group (JPG). To extract the textual information for using in a data mining pipeline, optical character recognition (OCR) engines have been developed for converting image data into machine-readable text. However, the accuracy of the OCR-generated text can be affected by many factors, such as algorithmic defects (e.g., segmentation, classification inaccuracy), limited hardware conditions (e.g., poor scanning equipment), and complex content status (e.g., a mixture of text fonts, complicated page layout) [10]. Although OCR engines include linguistic analysis in character recognition and segmentation [12], such analysis is limited and blind to the overall recognition performance.

In this paper, we overview our OCR post-processing system, which is an integrated correction engine that consumes OCR-generated text and produces its correction. The computational procedure can be conducted in a fully automatic manner, where optional interactive candidate selection can be integrated to further optimize the correction output. Utilizing web-scale linguistic corpora, our system conducts an elaborate candidate generation and a comprehensive linguistic analysis for correcting each error, and thus able to suggests high-quality candidate corrections that are orthographically similar to the error, consistent with the topic, and coherent to the context. We also optimize the processing flow for efficiently applying this model to real-world usage.

DocEng'17, September 4–7, 2017, Valletta, Malta.

DOI: http://dx.doi/org/10.1145/3103010.3121032

2 RELATED WORKS

OCR post-processing systems can be categorized as manual, semi-automatic, and automatic according to the degree of automation. A manual system that builts a full-text search tool to retrieve all occurrences of original images given a text query is introduced in [11]. It relies fully on the user to validate and correct OCR errors.

A number of studies view the post-processing of OCR output as the initial step in the error correction pipeline and involve continuous human intervention afterwards [13, 14]. 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 [13]. Further work records the previous human corrections to update the underlying Bayesian model for automatic correction [14].

One of the most recently proposed automatic systems, [4], made use of three n-gram statistical features extracted from three million documents to train a linear regressor for candidate ranking. Correction candidates suggested by this model are not restricted to those found in OCR 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.

A different direction of research uses an ensemble approach on the outputs of multiple OCR engines for the same input image and selects the best recognition for each word as the final output [5, 7–9]. Combining complementary recognition results from different OCR engines is claimed to lead to a better output [5]. The overall error rate is demonstrated to decrease with additional OCR engines involved, regardless of the performance of each added model in [9]. The application of both OCR recognition votes and lexical features to train a Conditional Random Field model and evaluate the test set in a different domain is proposed in [8]. While such models have proved useful, they select words only among OCR model recognitions and are blind to other candidate words. Besides, they require the presence of the original OCR input and effort of multiple OCR processing.

3 SYSTEM OVERVIEW

The post-processing workflow of our system, shown in Fig. 1, follows the conventional error correction steps: *error word detection*, *candidate correction generation* for the detected errors, and *candidate ranking* [6].

Our system applies web-scale linguistic corpora, including a word *n*-gram corpus and lexicons. A lexicon is a word list, which is interchangeable with "dictionary" in the literature. A word *n*-gram corpus is a collection of *n*-grams (i.e. *n* consecutive words in the text) with occurrence frequency in a document set, where some notable examples are Google Book *n*-gram [1] or Google Web 1T 5-gram corpus [2].

3.1 Computational Procedure

Text Preprocessing. To achieve a fully automatic text correction system, we integrate a text preprocessing step before error detection to handle potential formatting issues.

Word Segmentation. Words are segmented using Google *n*-gram tokenization[3], and over-segmented fragments are merged using heuristics. Unlike other correction models that split words by whitespace [6], we apply Google *n*-gram tokenization to generate more meaningful words, with rules that segment punctuations into separate tokens, disambiguate quotations, split hyphenated words, and separate contractions such as *she'll* into *she* and *'ll*. OCR error may contain mis-recognized punctuation, for example <*family* → *famil}^*>, in which a word is split into multiple tokens. Using heuristic rules designed for general noise texts, we automatically disambiguate such erroneous word fragments and merge them into one unit to detect.

Error Detection. Words and corresponding context *n*-grams are statistically analyzed for error word identification. A word is detected as an error if it fulfills one of the following two conditions: (1) its unigram frequency is less than a threshold, or (2) there is no context *n*-gram, with frequency in the corpus that exceeds another predefined threshold. According to Zipf's law [15], the word length correlates to the word frequency. We thus set up a unigram threshold that increases with the length of the detected word.

Candidate Generation. The candidate corrections for each error are merged from two types of generation algorithms: *reverse Levenshtein distance search* and *word n-gram context search*. Both algorithms search for candidates in the unigram corpus, where the former one selects candidates that are orthographically similar to the error word and the latter one prefers candidates that are coherent with the local context. When searching for candidates in word *n*-gram corpus, we also retrieve the corresponding statistics for use by subsequent contextual analysis in order to avoid repeated search in the large corpus.

Linguistic Analysis. Different measures are applied to quantitatively analyze the linguistic features for each generated candidate. For each candidate, the analysis results from different measures are collected. The candidate set that is considered in the subsequent steps is the union of the top-ranked candidates in each linguistic analysis.

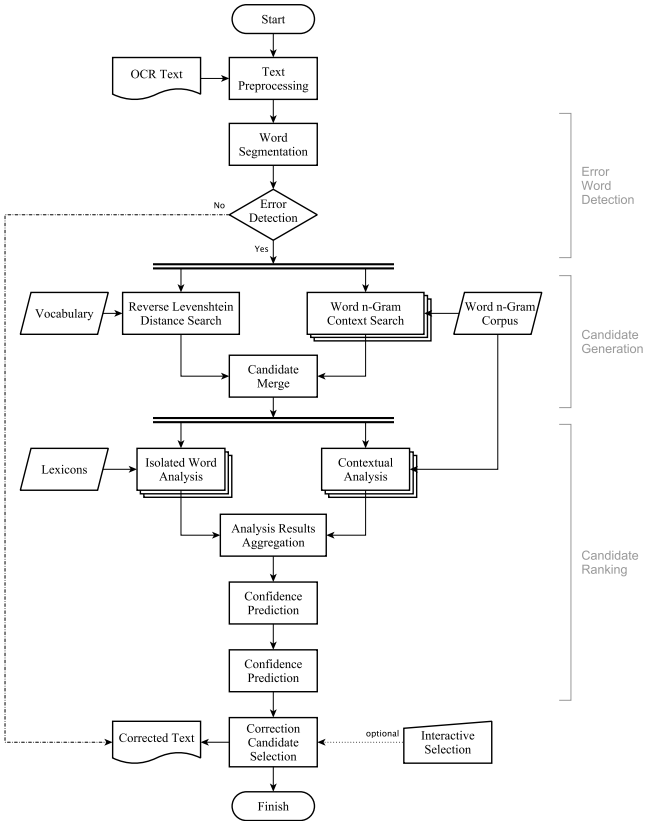

Figure 1: Process flowchart of the proposed OCR post-processing system.

Confidence Prediction. The confidence of each candidate being a valid correction is predicted by a supervised regressor given the candidate analysis result, which is used to rank among all candidates of an error.

Candidate Selection. The top-ranked correction candidates are automatically selected to substitute the error words in the text. In addition, an interactive candidate selection can be optionally adopted to optimize the selection among a ranked candidate list.

3.2 Linguistic Analysis

A list of the applied linguistic features is given in Table 1. According to the required textual information, we categorize the applied linguistic analysis in two types: (1) *isolated-word analysis* that uses the error word for candidate evaluation, and (2) *contextual analysis* that considers the context words. Besides, these two analysis types utilize different corpora, as shown in Fig. 1.

Edit Distance. Edit distance is a fundamental technique in quantifying the difference between two strings and is widely used for approximate string matching. We apply the unit-cost Levenshtein distance value to evaluate candidates.

[1] https://books.google.com/ngrams
[2] https://catalog.ldc.upenn.edu/ldc2006t13
[3] https://catalog.ldc.upenn.edu/docs/LDC2006T13/readme.txt

Table 1: System applied linguistic features.

Analysis Type	Linguistic Feature
Isolated-word Analysis	Levenshtein distance
	Lexical similarity
	Language Popularity
	lexicon existence
Contextual Analysis	Exact context coherence
	Relaxed context coherence

Lexical Similarity. While edit distance measures the differences between two words, it is blind to the similarity in two character sequences. Thus, we apply a lexical similarity measure, [3], which considers the common subsequences between two given strings.

Language popularity. We consider the adequacy of the suggested candidates to avoid selecting the uncommon words, which are measured by the candidate word frequency in a corpus.

Lexicon existence. We want highly ranked candidates to be semantically consistent with the text topic. The evaluation score for each topic is a boolean value that indicates the existence of the candidate in the according lexicon. With this type of feature, the model is able to estimate the topics of the training data and distinct the semantically related candidates in ranking.

Exact Context Coherent. An appropriate correction candidate should be coherent with the context where the error occurs. Using statistics from a web-scale 5-gram corpus, we evaluate suggested candidates their syntactic coherence with local context in text.

Relaxed Context Coherent. Consider there are limited candidates that can be suggested from the exact 5-gram context containing rare words, we relax the 5-gram matching condition by allowing mismatched word.

3.3 Processing Optimization

It is computationally intensive to adopt the proposed feature-based statistical approach to OCR Error correction. To leverage linguistic analysis in candidate ranking, the analytic result of each linguistic feature is computed for all generated candidates. While relaxing the generation criteria (i.e. increase the distance threshold in reversed Levenshtein distance or the number of relaxed words in context search) increases the probability of including the correction in the generated candidates, the number of candidates increases drastically and thus requiring more computation in analyzing their linguistic features. It is especially problematic when using web-scale corpora.

To facilitate parallel computing with MapReduce, we define analyzing batch as a series of analyzing jobs for the same linguistic feature of multiple words. In addition, we apply optimization strategies in processing multiple words within batch: For isolated-word analysis, we distinguish unique words in the analyzing batch before computation to avoid re-computation for error words with the same string representation. For contextual analysis, we first collect and distinguish unique 5-grams contexts from error words in batch. Consider 5-gram records in the corpus are ordered lexicographically, we cluster contexts according to their leading words and search each cluster in a bounded corpus region containing records with the same leading words.

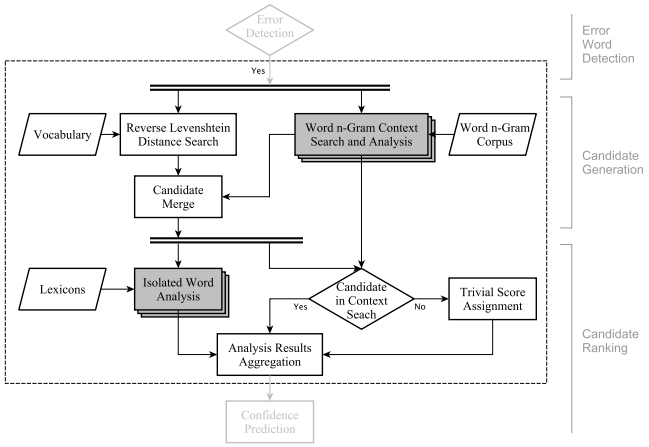

Figure 2: Process flowchart of the computational reduced candidate generation and linguistic analysis. Gray blocks are able to be processed in batch mode.

To further reduce expensive search operations in a web-scale 5-gram corpus, we modify the processing workflow of candidate generation and linguistics analysis to reduce computation as illustrated in Fig. 2. Since contextual analysis leverages frequency statistics in 5-gram corpus, a candidate receives nontrivial analytics score from a context feature only if it is generated from the 5-gram corpus and frequency statistics used for analyzing such candidate can be retrieved during search. Thus, we compute contextual features for candidates that are generated from the same feature and assign trivial contextual analytics scores for the rest candidates.

4 EVALUATION

We evaluate the system on an OCR text generated by Tesseract[4] from a book titled "Birds of Great Britain and Ireland, Vol. 2" [1]. This dataset contains listed OCR-generated errors along with the ground truth and OCR text for benchmark testing.[5] We investigate the following questions:

- How capable is each linguistic evaluation of distinguishing the intended correction within the top suggestions? How differently does each linguistic contribute to the candidate generation?
- How much can feature-based prediction optimize an automatic candidate selection?

4.1 Applied Linguistic Corpora

We use Google Web 1T 5-gram corpus, containing word 1-gram (unigram) to 5-gram, which is generated from approximately 1 trillion word tokens that extracted from publicly accessible web pages. Words recorded in the unigram corpus are used as the system vocabulary.

We use three lexicons: (1) *Wikipedia entities* extracted from page titles in Wikipedia, (2) *domain-specific glossaries* containing rare

[4]https://github.com/tesseract-ocr/tesseract
[5]https://github.com/jmei91/MiBio-OCR-dataset

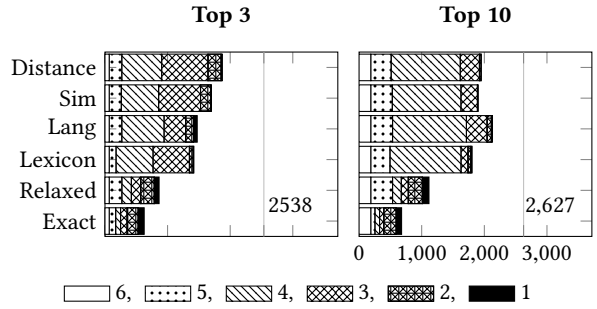

Figure 3: The coverage and distinctiveness of the linguistic features in locating the ground truth corrections.

terms in different domains, and (2) a *biodiversity terminology list* to fit the topic of the evaluation data.

4.2 Candidate Suggestion Evaluation

We evaluate the contribution of different linguistic evaluations on candidates generation. Given the top 10 candidates from each linguistic evaluation, Fig. 3 shows the number of intended corrections located by each evaluation. The distinctiveness levels of the discovered corrections are represented in colors, where the darker bar indicates the corresponding portion of candidates that are top-ranked in fewer features and vice versa. It shows that the lexical analysis using edit distance is the most effective way of identifying the intended correction, given the fact that a large portion of OCR errors are orthographically similar to their correction. The correction coverage can be further improved with other evaluation measures involved, where some corrections are uniquely suggested from word popularity analysis and contextual analysis. While the top candidates suggested from contextual analysis have limited correction coverage, they are able to find corrections that are uncommon and orthographically dissimilar, and thus important for improving the overall correction coverage.

We compare candidate generation with Aspell version 0.60.7. Aspell is an open-source error correction software preinstalled in Linux distributions. Although Aspell contains a comprehensive dictionary, it suggests candidates only within a limited Damerau-Levenshtein distance and thus only a small fraction of corrections, which are orthographically similar to the errors, are able to be find.

4.3 Candidate Ranking Evaluation

Our system in average generates more than 42,000 candidates for each error, given distance threshold 3 in reverse Levenshtein search and one mismatching word in n-gram context search. The intended corrections should be ranked close to the top. We evaluate the prediction accuracy given a ranked candidate list. Experimentally, we apply an AdaBoost.R2 [2] regressor on top of decision trees with the linear loss function. Table 2 shows that there are 61.05% of the OCR error can be corrected in a fully automatic manner. Involving user interaction for selection among top 10 candidates, the system accuracy can be improved to 76.62%, which is significantly higher than the accuracy from interactive mode supported by Aspell.

Table 2: Candidate ranking performance evaluation.

System	Proposed System				Aspell
P@n	1	3	5	10	
Precision	0.6105	0.7145	0.7378	0.7662	0.2583
Δ		+0.1040	+0.0233	+0.0284	

5 CONCLUSION

We have introduced an OCR post-processing system for automatically correcting OCR-generated errors in the text. Web-scale corpora are applied to candidate generation and linguistic analysis for each feature. By integrating different linguistic analysis results in a regression process, our system is able to select and rank high-quality candidates. Evaluated on an OCR-generated natural history book, our system can correct 61.5% of the errors in a fully automatic manner and 76.62% by interactive selection from among the top 10 candidates.

REFERENCES

[1] Arthur G. Butler, William Frohawk, Frederick, and H. GrÃűnvold. 1907. *Birds of Great Britain and Ireland. by Arthur G. Butler.* Vol. 2. Hull; Brumby & Clarke. 341 pages. http://www.biodiversitylibrary.org/item/35947#page/13/mode/1up

[2] Yoav Freund and Robert E Schapire. 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Comput. Syst. Sci.* 55, 1 (08 1997), 119–139. https://doi.org/10.1006/jcss.1997.1504

[3] Aminul Islam and Diana Inkpen. 2009. Real-word Spelling Correction Using Google Web 1Tn-gram Data Set. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM '09)*. ACM, New York, NY, USA, 1689–1692. https://doi.org/10.1145/1645953.1646205

[4] I. Kissos and N. Dershowitz. 2016. OCR Error Correction Using Character Correction and Feature-Based Word Classification. In *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*. 198–203. https://doi.org/10.1109/DAS.2016.44

[5] Shmuel T Klein, M Ben-Nissan, and M Kopel. 2002. A voting system for automatic OCR correction. (August 2002).

[6] Karen Kukich. 1992. Techniques for Automatically Correcting Words in Text. *ACM Comput. Surv.* 24, 4 (12 1992), 377–439. https://doi.org/10.1145/146370.146380

[7] William B. Lund and Eric K. Ringger. 2009. Improving Optical Character Recognition Through Efficient Multiple System Alignment. In *Proceedings of the 9th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '09)*. ACM, New York, NY, USA, 231–240. https://doi.org/10.1145/1555400.1555437

[8] William B. Lund, Eric K. Ringger, and Daniel D. Walker. 2013. How well does multiple OCR error correction generalize?. In *Proc. SPIE*, Vol. 9021. 90210A–90210A–13. https://doi.org/10.1117/12.2042502

[9] W. B. Lund, D. D. Walker, and E. K. Ringger. 2011. Progressive Alignment and Discriminative Error Correction for Multiple OCR Engines. In *2011 International Conference on Document Analysis and Recognition*. 764–768. https://doi.org/10.1109/ICDAR.2011.303

[10] Jie Mei, Aminul Islam, Yajing Wu, Abidalrahman Moh'd, and Evangelos E. Milios. 2016. Statistical Learning for OCR Text Correction. *CoRR* abs/1611.06950 (2016). http://arxiv.org/abs/1611.06950

[11] GÃijnter MÃijhlberger, Johannes Zelger, and David Sagmeister. 2014. User-driven Correction of OCR Errors: Combining Crowdsourcing and Information Retrieval Technology. In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage (DATeCH '14)*. ACM, New York, NY, USA, 53–56. https://doi.org/10.1145/2595188.2595212

[12] Ray Smith. 2007. An Overview of the Tesseract OCR Engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, Vol. 2. 629–633. https://doi.org/10.1109/ICDAR.2007.4376991

[13] Kazem Taghva, Julie Borsack, and Allen Condit. 1994. Expert system for automatically correcting OCR output. *Proc. SPIE* 2181, 270–278. https://doi.org/10.1117/12.171114

[14] Kazem Taghva and Eric Stofsky. 2001. OCRSpell: an interactive spelling correction system for OCR errors in text. *International Journal on Document Analysis and Recognition* 3, 3 (2001), 125–137. https://doi.org/10.1007/PL00013558

[15] George Kingsley Zipf. 1935. The psycho-biology of language. (1935).