Statistical Learning for OCR Text Correction

Jie Mei[1], Aminul Islam[2], Yajing Wu[1], Abidalrahman Moh'd[1], Evangelos E. Milios[1]

[1] Faculty of Computer Science, Dalhousie University
{jmei, yajing, amohd, eem}@cs.dal.ca

[2] School of Computing and Informatics, University of Louisiana at Lafayette
aminul@louisiana.edu

Abstract

The accuracy of Optical Character Recognition (OCR) is crucial to the success of subsequent applications used in text analyzing pipeline. Recent models of OCR post-processing significantly improve the quality of OCR-generated text, but are still prone to suggest correction candidates from limited observations while insufficiently accounting for the characteristics of OCR errors. In this paper, we show how to enlarge candidate suggestion space by using external corpus and integrating OCR-specific features in a regression approach to correct OCR-generated errors. The evaluation results show that our model can correct 61.5% of the OCR-errors (considering the top 1 suggestion) and 71.5% of the OCR-errors (considering the top 3 suggestions), for cases where the theoretical correction upper-bound is 78%.

1 Introduction

An increasing amount of data is produced and transformed into the digital form these days, including magazines, books, and scientific articles. Using the graphic formats, like Portable Document Format (PDF) or Joint Picture Group (JPG), is a comprehensive solution for efficient digitization as well as better preserving the page layout and the graphical information (i.e., charts and figures). Since information in such formats is not machine-readable, analyzing such data relies heavily on the accuracy of Optical Character Recognition (OCR) (Doermann, 1998). However, OCR systems are imperfect and prone to errors.

Post-processing is an important step in improving the quality of OCR output, which is crucial to the success of any text analyzing system in pipeline. An OCR Post-processing model attempts to detect misspellings in noisy OCR output and correct such errors to their intended representations. Many machine learning approaches (Lund and Ringger, 2009; Lund et al., 2011; Lund et al., 2013a; Lund et al., 2013b) correct the OCR-generated errors by selecting the most appropriate correction among candidates. OCR-generated errors are more diverse than handwriting errors in many aspects (Jones et al., 1991; Kukich, 1992b). Machine learning approaches incorporate different features enabling more robust candidate selection, instead of inferring from limited observations, for example, using a probabilistic-based model (Taghva and Stofsky, 2001).

While machine learning approach exhibits advantages in correcting OCR-generated texts, two problems emerge from the existing models: First, some models (Lund and Ringger, 2009; Lund et al., 2011; Lund et al., 2013a; Lund et al., 2013b; Kissos and Dershowitz, 2016) limit candidate suggestions from the recognition output of OCR engines. The errors unrecognized by all OCR engines are thus unable to be corrected. This issue can be problematic especially when original input suffers from degradation, for example, historical documents (Ntirogiannis et al., 2013). Secondly, another class of models (Kissos and Dershowitz, 2016) uses the frequencies of both candidate and related n-grams from corpus as features for training. Although n-gram statistics is shown to be effective in correcting real-word spelling errors (Islam and Inkpen, 2009b), training with only n-gram features does not capture the diverse na-

ture of OCR errors and may lead to a biased model where candidates with low frequency in the corpus tend to be not selected.

In this work, we propose an OCR post-processing error correction model that leverages different features through a learning process. Our model applies different features to avoid bias and improve the correction accuracy. To address the limitation of candidate suggestion, we enhance the scope of the candidates of an error by considering all the words available in the vocabulary within a limited Damerau-Levenshtein distance (Damerau, 1964) and then use features to narrow down the candidates number. The proposed model ranks the candidates by a regression model and shows that more than 61.5% of the errors can be corrected on a ground truth dataset. For 25.9% of the uncorrected errors, our model could provide the correction in top three suggestions.

To sum up, our contributions are as follows:

- We propose an OCR post-processing model which integrates OCR-specific features in a regression approach. The evaluation result shows that the proposed model is capable of providing high quality candidates in the top suggested list.

- We make available a ground truth OCR-error dataset, which is generated from a book in Biodiversity Heritage Library. This dataset lists the mappings from OCR-generated errors to their intended representations, which can be used directly for benchmark testing.

2 Related Works

The literature of OCR post-processing research exhibits a rich family of models for correcting OCR-generated errors. The post-processing model is an integrated system, which detects and corrects misspellings of both non-word and real-word in the OCR-generated text.

Some studies view the post-processing as the initial step in a correction pipeline and involve continuous human intervention afterwards (Taghva et al., 1994; Taghva and Stofsky, 2001; Mühlberger et al., 2014). These models are designed to reduce the human effort in correcting errors manually. Taghva et al. (1994) integrate dictionaries and heuristics to correct as many OCR errors as possible before these are given to human correctors. In their future work, Taghva and Stofsky (2001), record the previous human corrections to update the underlying Bayesian model for automatic correction. As an extreme case, Mühlberger et al. (2014) build a full-text search tool to retrieve all occurrences of original images given a text query, which fully relies on the user to validate and correct the errors.

One direction of work ensembles outputs from multiple OCR engines for the same input and selects the best word recognition as the final output (Klein et al., 2002; Cecotti and Belayd, 2005; Lund and Ringger, 2009; Lund et al., 2011; Lund et al., 2013a; Lund et al., 2013b). Klein et al. (2002) show that combining complementary result from different OCR models leads to a better output. Lund et al. (2011) demonstrate that the overall error rate decreases with the addition of different OCR models, regardless of the performance of each added model. Lund et al. (2013a) use machine learning techniques to select the best word recognitions among different OCR outputs. Lund et al. (2013b) apply both OCR recognition votes and lexical features to train a Conditional Random Field model and evaluate the test set in a different domain. While such models have proved useful, they select words only among OCR model recognitions and are blind to other candidates. Besides, they require the presence of the original OCR input and effort of multiple OCR processing.

Another class of post-processing models abstracts from OCR engines and leverages statistics from external resources (Bassil and Alwani, 2012a; Bassil and Alwani, 2012b; Kissos and Dershowitz, 2016). Kissos and Dershowitz (2016) use three n-gram statistical features extracted from three million documents to train a linear regressor for candidate ranking. Bassil and Alwani (2012a) make use of the frequencies in the Google Web 1T n-gram corpus (Brants and Franz, 2006) for candidate suggestion and ranking. Candidates suggested from these models are not restricted to exist in OCR recognitions. However, existing methods make use of solely n-gram frequencies without knowing the characteristics of OCR errors and are, thus, bias to select common words from the

n-gram corpus.

3 Characteristics of OCR Errors

The word error rate of OCR engines, in practice, is in the range of 7-16% (Santos et al., 1992; Jones et al., 1991), which is significantly higher than the 1.5-2.5% for Handwriting (Wing and Baddeley, 1980; Mitton, 1987) and the 0.2-0.05% for the edited newswire (Pollock and Zamora, 1984; Church and Gale, 1991). OCR-generated errors tend to have some distinct characteristics, which require different techniques than spell correction:

Complex non-standard edits The human-generated misspellings are character-level edits, which can be categorized into one of the following four standard types: *insertion*, *deletion*, *substitution*, and *transposition*. The majority of spell correction errors, roughly 80%, is single edit from the intended word (Damerau, 1964) and tend to be within one length difference (Kukich, 1992b). However, a significant fraction of OCR-generated errors are not one-to-one character-level edit (e.g., *ri* $\rightarrow$ *n* or *m* $\rightarrow$ *iii*) (Jones et al., 1991).

Multi-factor error generation OCR errors are generated in different processing steps due to various factors. Taghva and Stofsky (Taghva and Stofsky, 2001) trace the errors associated with the primary OCR steps involved in the conversion process: (1) *scanning error* caused by the low paper/print quality of the original document or the pool condition of the scanning equipment. (2) *zoning error* caused by incorrect decolumnization or complex page layout. (3) *segmentation error* caused by the broken characters, overlapping characters, and nonstandard fonts in the document. (4) *classification error* caused by the incorrect mapping from segmented pixels to a single character.

Multi-source dependent The characteristics of OCR-generated errors vary according to not only human reasons (e.g., publishers or authors) but also non-human causes (e.g., text font or input quality) (Jones et al., 1991). These are especially sensitive between OCR engines. Because different OCR engines use different techniques and features for recognition leads to a different confusion probability distribution (Kukich, 1992b).

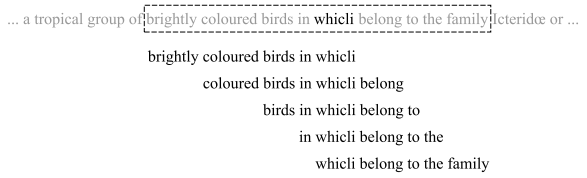

Figure 1: Context collection example for token "whicli" in an OCR-generated text. Using a sliding window of size five, we are able to construct five 5-gram contexts for "whicli".

4 Proposed Model

In this section, we describe in detail the processing steps of the proposed model. We use external resources during the correction process including lexicons and a word n-gram corpus. A lexicon[1] is a list of unique words and word n-gram corpus refers to a list of n-grams (i.e., n consecutive words) with observed frequency counts. Some examples of word n-gram corpus are Google Book n-gram (Michel et al., 2011) and Google Web 1T 5-gram corpus (Brants and Franz, 2006).

To annotate, we denote English strings with text font (e.g., w_c, s), vectors with bold lowercase (e.g., $\mathbf{x}$), sets with cursive uppercase (e.g., $\mathcal{C}, \mathcal{E}$), scalar with lower-case English or Greek characters (e.g., y, α), and functions followed by a bracket (e.g., $dist(*)$, $score(*)$). The size of a collection is represented as $||*||$ (e.g., $||\mathcal{D}||$).

4.1 Error Detection

Error detection step identifies errors in the tokenized text, which is the first step in the correction procedure. Since a correct word will not proceed to the further correction steps, we want to set a weak detection restriction to filter only highly confident words. We rely on the n-gram frequency to determine the correctness of a word. A word is detected as an error if any one of the following conditions does not fulfill.

- Consider a common word is less likely to be an error word, the 1-gram frequency of a word should be greater than a frequency threshold. The frequency threshold varies with different word length.

[1]The term "lexicon" is usually used interchangeably with "dictionary" and "word list" in the literature.

- A word is likely to be correct if this word with its context occurs in other places. We use a sliding window to construct n-gram contexts for a word. The frequency of one of the context in the n-gram corpus should be greater than a frequency threshold.

4.2 Candidate Search

We select a candidate set for each error, which contains all the words in the vocabulary within a limited number of character modifications. To be specific, let Σ be the symbol set, $\mathcal{L} \in \Sigma^*$ be a language lexicon. The candidate set for a detected error w_e is:

$$\{ w_c \,|w_c \in \mathcal{L}, dist(w_c, w_e) \leq \delta\}, \qquad (1)$$

where $dist(*)$ is the minimum edit distance and δ is a distance threshold. Damerau-Levenshtein distance (Damerau, 1964), which is used in most of the spell correction models for locating the candidates, considers all four character-level editing types (mentioned in Section 3). Since transposition errors are common in human-generated text but rarely occur in the OCR-generated text, we apply Levenshtein distance (Levenshtein, 1966), which uses a simpler operation set without transposition.

4.3 Feature Scoring

We score each error candidate by features. In this section, we discuss the contribution of the features in candidate estimation and describe the scoring measures applied in our model.

Levenshtein edit distance Minimum edit distance is a fundamental technique in quantifying the difference between two strings in spell correction. Given two string s_1 and s_2 on alphabet Σ, the edit distance $dist(\mathrm{s}_1, \mathrm{s}_2)$ is the minimum number of edit operations required to transform from s_1 into s_2 (Wagner, 1974). An edit operation is a character-level modification in Σ. We use Levenshtein edit distance for the same reason as previously described in Section 4.2. The score function is as follows:

$$score(\mathrm{w}_c, \mathrm{w}_e) = 1 - \frac{dist(\mathrm{w}_c, \mathrm{w}_e)}{\delta + 1} \qquad (2)$$

String similarity Longest common subsequence (Allison and Dix, 1986) (LCS) is an alternative approach than edit distance in matching similar strings. There are variations of LCS: *Normalized Longest Common Subsequence (NLCS)*, which take into account the length of both the shorter and the longer string for normalization.

$$nlcs(\mathrm{w}_c, \mathrm{w}_e) = \frac{2 \cdot len(lcs(\mathrm{w}_c, \mathrm{w}_e))^2}{len(\mathrm{w}_c) + len(\mathrm{w}_e)}. \qquad (3)$$

Normalized Maximal Consecutive Longest Common Subsequence (MCLCS), which limits the common subsequence to be consecutive. There are three types of modifications with different additional conditions: *NLCS*$_1$ and *NLCS*$_n$ use the subsequences starting at the first and the n-th character, respectively; *NLCS*$_z$ takes the subsequences ending at the last character. They apply the same normalization as *NLCS*.

$$nmnlcs_1(\mathrm{w}_c, \mathrm{w}_e) = \frac{2 \cdot len(mclcs_1(\mathrm{w}_c, \mathrm{w}_e))^2}{len(\mathrm{w}_c) + len(\mathrm{w}_e)} \qquad (4)$$

$$nmnlcs_n(\mathrm{w}_c, \mathrm{w}_e) = \frac{2 \cdot len(mclcs_n(\mathrm{w}_c, \mathrm{w}_e))^2}{len(\mathrm{w}_c) + len(\mathrm{w}_e)} \qquad (5)$$

$$nmnlcs_z(\mathrm{w}_c, \mathrm{w}_e) = \frac{2 \cdot len(mclcs_z(\mathrm{w}_c, \mathrm{w}_e))^2}{len(\mathrm{w}_c) + len(\mathrm{w}_e)}. \qquad (6)$$

We apply the measure proposed in (Islam and Inkpen, 2009a) for scoring, which takes the weighted sum of the above LCS variations:

$$\begin{aligned} &score(\mathrm{w}_c, \mathrm{w}_e) \\ &= \alpha_1 \cdot nlcs(\mathrm{w}_c, \mathrm{w}_e) + \alpha_2 \cdot nmnlcs_1(\mathrm{w}_c, \mathrm{w}_e) \\ &+ \alpha_3 \cdot nmnlcs_n(\mathrm{w}_c, \mathrm{w}_e) + \alpha_4 \cdot nmnlcs_z(\mathrm{w}_c, \mathrm{w}_e). \end{aligned} \qquad (7)$$

Language popularity Using a language lexicon is a common approach to detect the non-word tokens, where non-existing tokens are detected as true errors. Let w_c be the candidate string, $\mathcal{C}$ be the set of all error candidates, and $freq_1(\cdot)$ be the unigram frequency. The candidate confidence is the unigram popularity given by:

$$score(\mathrm{w}_c, \mathrm{w}_e) = \frac{freq_1(\mathrm{w}_c)}{\max_{\mathrm{w}'_c \in C} freq_1(\mathrm{w}'_c)}. \qquad (8)$$

Lexicon existance Besides English lexicon, we can use different lexicons to detect the existence of the token in different subjects. It identifies additional lexical features. For example, we may use a domain specific lexicon to capture terminologies, which is especially useful for input text from the same domain. The candidate selection is the same as English lexicon, but the candidate score is a boolean value that indicates the detection result.

$$score(\mathrm{w}_c, \mathrm{w}_e) = \begin{cases} 1 & \text{if } \mathrm{w}_c \text{ exists in the lexicon} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Exact-context popularity An appropriate correction candidate should be coherent in context. Using word n-gram for context analysis is a broadly researched approach in correcting real-word errors (Islam and Inkpen, 2009a). Given an error word w_e in a text, we have its n-gram contexts $\mathcal{G}$ constructed using a sliding window (see Figure 1). To score a candidate w_c of this error, we first substitute the error word from each of its n-gram contexts by such candidate and create a new set of contexts $\mathcal{G}_c$. Let $\mathcal{C}$ be all candidates suggested for w_e, and $freq_n(\cdot)$ be the n-gram frequency, which gives 0 to a non-existing n-gram. The score function is given as:

$$score(\mathrm{w}_c, \mathrm{w}_e) = \frac{\sum_{\mathbf{c} \in \mathcal{G}_c} freq_n(\mathbf{c})}{\max_{\mathrm{w}'_c \in \mathcal{C}} \{\sum_{\mathbf{c}' \in \mathcal{G}'_c} freq_n(\mathbf{c}')\}} \tag{10}$$

Relaxed-context popularity A context with longer n-gram size defines a more specific use case for a given word, where its existence in the corpus shows higher confidence for a candidate. In general, an n-gram corpus has limited coverage for all possible n-grams in the language, especially for the emerging words in the language. Candidates of a rare word can barely be suggested from its contexts because of the limited coverage in the n-gram corpus. We deal with such issue by relaxing the context matching condition to allow one mismatching context word. For example, in Figure 1, we consider only the first 5-gram context given "which" be the candidate. We need the frequency of "brightly coloured birds in which" for computing exact context popularity. As for the relaxed context popularity, we need to sum up the frequencies of four types of 5-grams: "* coloured birds in which", "brightly * birds in which", "brightly coloured * in which", and "brightly coloured birds * which", where * matches any valid unigram. The scoring function is the same as the exact context matching (Eq. 10), except for the candidate set and the context set are larger in the relaxed case.

4.4 Candidate Ranking

We formulate the confidence prediction task as a regression problem. Given candidate feature scores, we predict the confidence of each candidate being a correction for the error word. The confidence is used for ranking among candidates of one error.

To train a regressor for correction, we label candidate features with 1 if a candidate is the intended correction, or 0 otherwise. The training data contains candidates from different errors, and there are more candidates labeled 0 than 1. To deal with the unbalanced nature of the candidates, we weight the samples when computing the training loss

$$loss(\mathcal{D}) = \sum_{e \in \mathcal{E}} \sum_{c \in \mathcal{C}_e^{\mathcal{F}}} w_c \cdot loss(\mathbf{x}_c, y_c). \tag{11}$$

We count the number of samples with label 1 and 0, respectively. Then, we use the ratio to weight for samples labeled 1, and 1 for samples labeled 0.

Experimentally, we apply a AdaBoost.R2 (Freund and Schapire, 1997) model on top of decision trees with linear loss function.

5 Evaluation

To better describe the error sample, we use the annotation $<\mathrm{w}_t \rightarrow \mathrm{w}_e>$ to represent the intended word w_t being recognized as the error word w_e.

5.1 Experimental Dataset

We made available a dataset with 2728 OCR-generated errors along with the ground truth and OCR text for benchmark testing. The OCR text was generated from the book titled "Birds of Great Britain and Ireland" (Butler et al., 1907) and made it publicly available by the Biodiversity Heritage Library (BHL) for Europe[2]. The ground truth text

[2] http://www.biodiversitylibrary.org/item/35947#page/13/mode/1up

Table 1: The Levenshtein edit distance distribution of error in the experimental dataset.

Levenshtein edit distance	Error Statistics		Sample Error	
	Number	Percentage	Intended Word	Error Word
1	669	24.60%	galbula	ga/bula
2	1353	49.76%	yellowish	jˆellowish
3	296	10.89%	bents	Ijcnts
4	163	5.99%	my	ni}’
5	82	3.02%	Lanius	Lioiiits
6	52	1.91%	minor	)iii>iof
7	29	1.07%	garrulus	f;ay>///us
8	19	0.70%	curvirostra	iUi’7’iyosira
9	9	0.33%	Nucifraga	Aiiii/rutˆd
≥ 10	26	0.96%	pomeranus	poiiui-iVtiis
total	2698	100%		

is based on an improved OCR output[3] and adjusted manually to match with the original content of the whole book.

This source image data of the book contains 460 page-separated files, where the the main content is included in 211 pages. This book combines different font types and layouts in main text, which leads erroneous OCR results. There are 2698 mismatching words between the ground truth text with the BHL digital OCR-text, which are used as the ground truth errors. The ground truth text contains 84492 non-punctuation words. Thus, the OCR error rate of the evaluation dataset is 3.22%, where some errors are complex regarding edit distance, shown in Table. 1. Other challenges include terminologies in multilingual (e.g., Turdidæ, Fringillidæ) and meanless words (e.g., bird-sound simulation: “cir-ir-ir-ir-re”, “vee-o”), which may not be handled using the standard techniques.

5.2 Evaluation Setup

Walker and Amsler (2014) claim that the lexicon from a published dictionary has limited coverage on newswire vocabulary, and vice versa. Thus, we construct a language lexicon with unigrams in the Google Web 1T n-gram corpus[4]. This corpus contains the frequencies of unigrams (single words) to five-grams, which is generated from approximately 1 trillion word tokens extracted from publicly accessible Web pages. Its unigram corpus is filtered with the frequency no less than 200. We use five-grams in Google Web 1T corpus for exact and relaxed context matching.

For lexicon existence feature, we use three lexicons to build two features instances: (1) *Wikipedia entities* extracted from article names in Wikipedia. This feature gives credit to common terminologies. (2) *Biodiversity terminologies* collected from biodiversity digital library to capture the domain specific terms, which may not be contained in Wikipedia.

The proposed model receives OCR-generated plain text as input. We apply the Penn Treebank tokenization with the additional rules from Google[5] to tokenize the input text. This tokenization method is consistent with the Google Web 1T n-gram corpus. The frequency and existence of rarely hyphenated words can be poorly estimated using external resources. Thus we split the hyphenated word by the internal hyphen.

Experimentally, we filter tokens of the following types after tokenization: (1) *punctuations*; (2) *numeric tokens*, which contains only numeric characters (i.e., 0-9); (3) *common English words*. We apply a lexicon of frequent English words for filtering. The accuracy of the system will increase with more relaxed filtering conditions on English words, for example, filtering only English stop

[3]http://www.bhle.eu/en/results-of-the-collaboration-of-bhl-europe-and-impact

[4]https://catalog.ldc.upenn.edu/LDC2006T13

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

Table 2: Confusion matrix for error detection

		Model Detection: Error	Model Detection: Correct	total
Actual Correctness	Error	2457	241	2698
	Correct	1273	80523	81794
	total	3730	80764	

words or even no filtering, but the computation time increases as the trade-off. Similarly for reducing the candidate detection time in Eq. 1, we set the maximum Levenshtein distance δ for candidate search to be 3.

5.3 Detection Evaluation

We evaluate error detection as a recall oriented task, which focus on finding all possible errors. In all error correction techniques, an undetected error will not get into the correction phase.

We report the confusion matrix for error detection in Table 2. The proposed model achieves 91.07% detection recall. There are considerable number of ture-positive errors, which are correct words but detected as errors. When using this type of errors for training or testing, we use the word itself as the intended word for each error. The correction results regarding to all types of errors are reported in Section 5.4.

For tokenizing the noisy text, any tokenization approach is inevitably involved in the common word boundary problem (Kukich, 1992b), the correct boundary of the errors are not properly identified, in both human-generated (Kukich, 1992a) and OCR-generated text (Jones et al., 1991). Such problem can be caused by the splitting (e.g., <*spend→sp end*>) and merging (e.g., <*in form→infrom*>) mistakes. It is especially problematic in OCR-generated text, where words containing characters are recognized as punctuation and are thus splitted by the tokenization heuristics. Most error detection and correction techniques define token as character sequence separated by white space characters (e.g., blanks, tabs, carriage returns, etc.) (Kukich, 1992b), which do not split the error token by punctuations. However, this approach cannot distinguish between true punctuation and misrecognized trailing punctuation (e.g., <*family→famil*}ˆ>).

Table 3: The number of detected errors and recall of bounded and unbounded detections

Detection Category	Number	Recall
Bounded	1995	73.94%
Unbounded	462	17.12%
Total (True-Positive)	2457	91.07%

An error may be "partially" detected if an overlapped but non-identical character sequence is treated as an error. We call this "partially" detected case as a success *unbounded* detection, where the correct recognition of the character sequence as success *bounded* detection. Unbounded detection can potentially be corrected, but it has inaccurate features scores that will influence the correction accuracy. For example, if an error <*spend→sp end*> is unbounded and detected as *end*, *sp* will exists in the context and candidate edit distance will be computed with *end* instead of *sp end*. In addition, there may exist multiple unbounded errors detected for one ground truth error, because of the splitting mistakes. For every ground truth error, we count at most one successful unbounded detection. Our model achieves the 73.51% bounded detection recall and 90.51% total detection recall (i.e., sum of bounded and unbounded detection) shown in Table 3.

5.4 Correction Evaluation

We take the following steps to build a training dataset: First, we construct a candidate set for each error containing top 10 candidates scored by each feature. Then, we select a subset of errors, whose intended word exists in the candidate set. Finally, we randomly select 80% errors and use their candidates sets for training.

We train multiple AdaBoost regressors with different settings and apply 10-fold cross-validation to select the best setting for evaluating the rest errors. We report the correction results regarding different error categories in Table 4. *P@n* represents precision at top n candidate suggestions,

Table 4: The percentage of errors, where correction exists among top 1, 3, 5, and 10 candidates suggested by the proposed model.

Error Categories	P@1	P@3	P@5	P@10
Bounded	0.6369	0.7710	0.8028	0.8405
Unbounded	0.5637	0.6417	0.6620	0.6823
True-Positive	0.6095	0.7025	0.7238	0.7604
False-Positive	0.6971	0.7145	0.7738	0.7942
Total	0.6150	0.7145	0.7378	0.7662

Table 5: The number and the percentage of errors, where correction exists among the top 10 candidates of any applied feature.

Detection Category	Correct Candidates		
	Number	Percentage Among All	Percentage in Search Scope
Bounded	1540	77.19%	84.71%
Unbounded	108	23.38%	47.58%
True-Positive	1648	67.07%	78.66%
False-Positive	1273	100.00%	100.00%
Total	2627	66.15%	78.00%

which calculate the ratio of the existence of intended words in top n candidates. The proposed model rank the candidates by a regression model and show that more than 61.5% of the errors can be corrected. For 25.9% of the uncorrected errors, our model could provide the correction in the top three suggestions.

6 Discussion

6.1 Selected Features

We want to study the contribution of features to candidate suggestion. We first explore how well the scoring functions could rank the intended words to the top without predicting by the regressor. For each detected error, we construct a candidate set containing top 10 candidates scored by each feature and check whether this candidate set contains a correction. Note that candidate search scope is limited by the number of edit distance δ (in Eq. 1 by default), thus the intended words w_t for w_e cannot be found if $dist_{lev}(\mathrm{w}_t, \mathrm{w}_e) > \delta$. Results are shown in Table 5. The model could locate most of the correction in top candidates with the collaboration of all applied features. We observe that performance varies drastically for bounded and unbounded errors, presumably because the feature score for unbounded errors is inaccurate (e.g., the split part of a splitting error is counted as the context word in context search).

Table 6: The percentage of errors, where correction exists among top 1, 3, 5, and 10 candidates suggested by Support Vector regressor (SVR), Rigid Linear regressor (RL), Multiple Layer Perceptron with rectified linear unit (MLP.ReLU), Random Forest (RF), and AdaBoost.R2

Regression Model	All Errors			
	P@1	P@3	P@5	P@10
SVR	0.5634	0.7158	0.7378	0.7612
RL	0.5637	0.6817	0.7020	0.7313
MLP.ReLU + BFGS	0.6095	0.7025	0.7283	0.7604
RF	0.6071	0.7145	0.7378	0.7516
AdaBoost.R2 + DT	0.6150	0.7145	0.7378	0.7662

To get a better intuition for the contribution of individual feature, we plot the distinctiveness of the located error corrections by each feature in Figure 2. For bounded detected errors, context-based features are able to locate some distinctive corrections, which can rarely be found by other features. In addition, Relax context popularity feature shows better coverage than exact context popularity. On the other hand, the other four features are important for false-positive errors, where context-based features provide little help.

6.2 Regression Model Selection

We report candidate ranking performance of different regression models in Table 6. The same training and testing dataset, described in Section 5.4, are used for all models.

Given the upperbound of correction rate within three edit distance is 78% (in Table 5), all regressors achieve good results. As can be seen, ensemble methods, like Random Forest and AdaBoost, are more robust than others in suggesting appropriate candidates.

7 Conclusion

We introduce a statistical learning model for correcting OCR-generated errors. By integrating different features in a regression process, our model

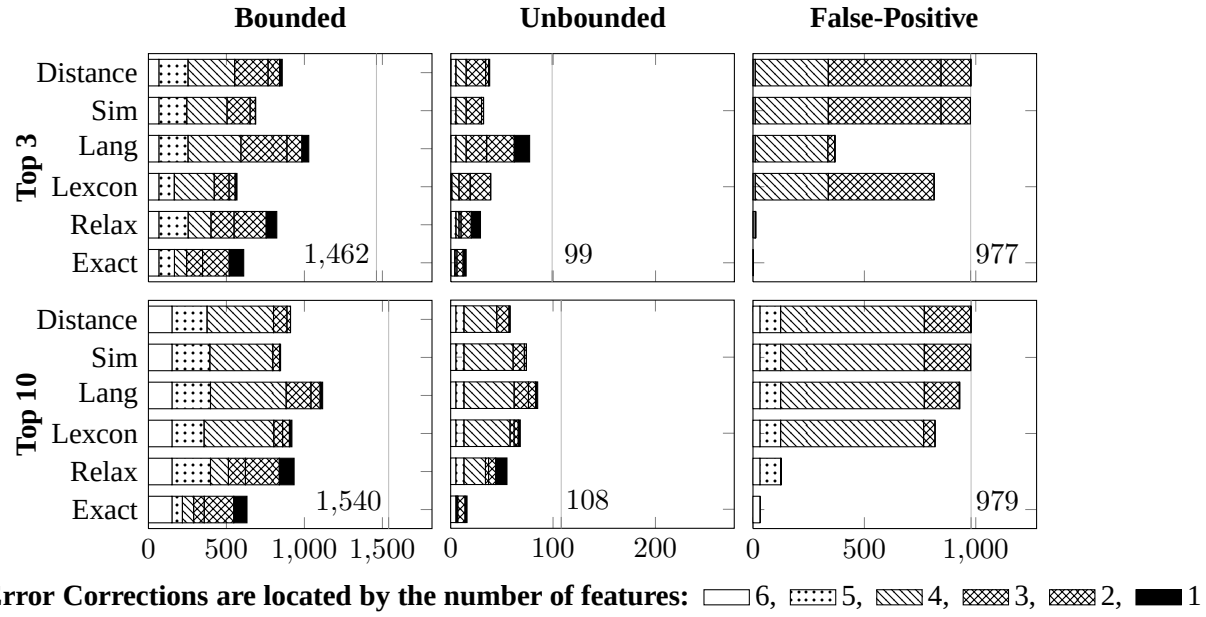

Figure 2: The distinctiveness of features in locating error corrections. A bar of a feature represents the number of error corrections located by this feature. The color of a bar indicates the number of features that locates these errors. i.e., white bar indicates a portion of error corrections located by all the features, while black bar indicates error corrections located by only one feature.

is able to select and rank candidates that are similar in shape to the error, suitable for the domain, and coherent to the context. The evaluation results show that our model can correct 61.5% of the errors and could provide a correction in top three suggestions for 25.9% of the uncorrected errors. That is, by suggesting three candidates for each error, our model can correct 71.5% error cases in a theoretical correction upperbound of 78%.

References

L Allison and T I Dix. 1986. A bit-string longest-common-subsequence algorithm. *Inf. Process. Lett.*, 23(6):305–310, December.

Youssef Bassil and Mohammad Alwani. 2012a. Context-sensitive spelling correction using google web 1t 5-gram information. *Computer and Information Science*, 5(3):37–48.

Youssef Bassil and Mohammad Alwani. 2012b. Context-sensitive spelling correction using google web 1t 5-gram information. *CoRR*, abs/1204.5852.

Thorsten Brants and Alex Franz. 2006. Web 1t 5-gram version 1 ldc2006t13.

Arthur G. Butler, William Frohawk, Frederick, and H. Grönvold. 1907. *Birds of Great Britain and Ireland. by Arthur G. Butler ; illustrated by H. Gronvold and F.W. Frohawk. Order Passeres /*, volume 2. Hull; Brumby & Clarke.

Hubert Cecotti and Abdel Belayd. 2005. Hybrid ocr combination approach complemented by a specialized icr applied on ancient documents. In *Proceedings of the Eighth International Conference on Document Analysis and Recognition*, ICDAR '05, pages 1045–1049, Washington, DC, USA. IEEE Computer Society.

Kenneth W. Church and William A. Gale. 1991. Probability scoring for spelling correction. *Statistics and Computing*, 1(2):93–103.

Fred J. Damerau. 1964. A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7(3):171–176, March.

David Doermann. 1998. The indexing and retrieval of document images. *Comput. Vis. Image Underst.*, 70(3):287–298, June.

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):119–139, August.

Aminul Islam and Diana Inkpen. 2009a. 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, pages 1689–1692, New York, NY, USA. ACM.

Aminul Islam and Diana Inkpen. 2009b. Real-word spelling correction using google web it 3-grams. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, pages 1241–1249, Stroudsburg, PA, USA. Association for Computational Linguistics.

M. A. Jones, G. A. Story, and B. W. Ballard. 1991. Interating multiple knowledge sources in a bayesian ocr post-processor. *International Journal on Document Analysis and Recognition*, pages 925–933.

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)*, pages 198–203, April.

Shmuel T Klein, M Ben-Nissan, and M Kopel. 2002. A voting system for automatic ocr correction. August.

Karen Kukich. 1992a. Spelling correction for the telecommunications network for the deaf. *Commun. ACM*, 35(5):80–90, May.

Karen Kukich. 1992b. Techniques for automatically correcting words in text. *ACM Comput. Surv.*, 24(4):377–439, December.

V. I. Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, February.

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, pages 231–240, New York, NY, USA. ACM.

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*, pages 764–768, Sept.

William B. Lund, Douglas J. Kennard, and Eric K. Ringger. 2013a. Combining multiple thresholding binarization values to improve ocr output.

William B. Lund, Eric K. Ringger, and Daniel D. Walker. 2013b. How well does multiple ocr error correction generalize? volume 9021, pages 90210A–90210A–13.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, Steven Pinker, Martin A. Nowak, and Erez Lieberman Aiden. 2011. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176–182.

Roger Mitton. 1987. Spelling checkers,spelling correctors and the misspellings of poor spellers. *Inf. Process. Manage.*, 23(5):495–505, September.

Günter Mühlberger, 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, pages 53–56, New York, NY, USA. ACM.

Konstantinos Ntirogiannis, Basilios Gatos, and Ioannis Pratikakis. 2013. Performance evaluation methodology for historical document image binarization. *IEEE Trans. Image Processing*, 22(2):595–609.

Joseph J. Pollock and Antonio Zamora. 1984. Automatic spelling correction in scientific and scholarly text. *Commun. ACM*, 27(4):358–368, April.

Paulo J. Santos, Amy J. Baltzer, Albert N. Badre, Richard L. Henneman, and Michael S. Miller. 1992. On handwriting recognition system performance: Some experimental results. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 36(4):283–287.

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):125–137.

Kazem Taghva, Julie Borsack, and Allen Condit. 1994. Expert system for automatically correcting ocr output. volume 2181, pages 270–278.

Robert A. Wagner. 1974. Order-n correction for regular languages. *Commun. ACM*, 17(5):265–268, May.

Donald E Walker and Robert A Amsler. 2014. The use of machine-readable dictionaries in sublanguage analysis. *Analyzing Language in Restricted Domains: Sublanguage Description and Processing*, page 69.

Alan M Wing and Alan Baddeley, 1980. *Spelling errors in handwriting: a corpus and a distributional analysis.*, pages 251–285. Academic Press.