Post-OCR Error Detection by Generating Plausible Candidates

Thi-Tuyet-Hai Nguyen*, Adam Jatowt†, Mickael Coustaty*, Nhu-Van Nguyen* and Antoine Doucet*
*L3i, University of La Rochelle
Email: {hai.nguyen, mickael.coustaty, nhu-van.nguyen, antoine.doucet}@univ-lr.fr
†Graduate School of Informatics, Kyoto University
Email: adam@dl.kuis.kyoto-u.ac.jp

***Abstract*—The accuracy of Optical Character Recognition (OCR) technologies considerably impacts the way digital documents are indexed, accessed and exploited. Post-processing approaches detect and correct remaining errors to improve the quality of OCR texts. However, state-of-the-art approaches still need to be improved. Most of the existing post-OCR techniques use predefined error position lists or apply simple techniques to detect errors. In this paper, we describe a novel error detector using different features from character-level (including character noisy channel, index of peculiarity) to word-level (such as frequencies of n-grams, skip-grams, part-of-speech) Experimental results show that our approach outperforms the best performing techniques in the ICDAR 2017 Competition on Post-OCR text correction.**

***Keywords*-OCR error detection, post-OCR, OCRed text**

I. Introduction

In an effort to preserve and enable easy access to ancient documents, Optical Character Recognition (OCR) techniques have been continuously developed to transform paper-based documents into digital ones. However, various layouts and poor physical quality of degraded documents pose significant challenges to OCR engines. The last step in the OCR pipeline, OCR post-processing, makes an attempt to improve the quality of OCR documents by detecting and correcting errors.

OCR post-processing consists of two parts, *error detection* and *error correction*. Most of the prior post-processing approaches focus on the correction part; they use a predefined list of error positions or they detect errors in fairly simple ways. Nonetheless, it does not mean that the error detection part is not important as naturally one cannot correct errors without knowing their positions. In this paper, we focus on the error detection task.

Based on the kinds of errors to be detected, error detection approaches can be divided into two main types: *non-word error detection* and *real-word error detection* [1]. Note that there are also detection approaches without the explicit focus on particular types of errors, which we regard as the third type, *mixed error detection*. The first type simply detects misspelled tokens by using character n-gram analysis and lexicon look-up therefore, it cannot detect errors related to the context. The second type allows to overcome the context mismatch problem by taking into account word n-gram frequency, part-of-speech and semantic information. Finally, approaches under the *mixed error detection* type apply the same method for both the error types.

Our approach belongs to the third type. The main idea is that an OCR token proves itself as the correct one in view of its feature values computed and normalized by the highest value of the plausible candidate set. Besides common features for statistical error detection (word, n-grams, skip-grams, part-of-speech), we suggest two novel features including an adapted version of the index of peculiarity [2] and a frequency of an OCR token in its candidate sets. Experimental results reveal that our method reaches higher performance on the English datasets than the top performing teams of the ICDAR 2017 competition on Post-OCR text correction [3].

The remainder of this paper is organized as follows. Section II discusses the related work. Details of our approach and the experimental results are described in Section III and IV, respectively. Finally, we conclude the paper and outline our future work in Section V.

II. Related work

This section discusses the three main types of error detection approaches: *non-word error detection*, *real-word error detection*, and *mixed error detection*.

A. Non-word error detection

Approaches of *non-word error detection* type concentrate on character frequencies and dictionary entries, therefore they cannot deal with context-sensitive errors.

1) Character n-gram analysis: Various forms of character n-gram tables (from non-positional to positional table, or from binary to frequency table) have been employed to detect errors. Zamora *et al.* [4] reported that the frequency trigram table was ineffective in distinguishing between valid/invalid words. Morris *et al.* [2] examined the frequency n-gram table from the document itself. They indicated that misspellings often contained trigrams which were peculiar to the checked document. Moreover, they suggested a fomular to calculate the index of peculiarity of each token.

2) Lexicon look-up techniques: Lexicon look-up techniques check each token of the input text in a lexicon. If the token does not match any lexicon entries, it is flagged as an error. However, a lexicon needs to be carefully

tuned to the application domain. Too small a lexicon can cause many false rejections of valid tokens. In contrast, too large a lexicon may contribute to a high number of false acceptances.

One of the ICDAR 2017 competition's approaches, (EFP) [3], explored lexicon look-up techniques and regular expression to preprocess and detect errors.

B. Real-word error detection

The second type overcomes the problems of the *non-word error detection* type by considering word context, such as Part-of-Speech (POS).

A method called *Trigrams* [5] encoded the context by part-of-speech (POS) trigrams. This method replaced an OCR token by each of its candidates. The probability of the resulting phrase was computed for each substitution. The candidate with the highest probability was suggested as the correction. However, this method cannot distinguish among words with the same POSs.

C. Mixed error detection

Approaches of this type apply the same techniques to detect *non-word* and *real-word* errors. Some prior approaches utilized frequencies of word and word ngrams to detect errors. A token is viewed as an error if its frequency or its ngram frequencies are less than a threshold [6]. Similarly, Khirbat [7] combined ngram frequencies with a presence of non alphanumeric text within a token to classify whether the token is an error or not.

Using word frequency, Taghva *et al.* [8] divided words into two sets: highly frequent words (centroids) and the rest (centroids' complement), then clustered centroids complement according to centroids using local information of checked documents and confusion matrix. A word which is a centroids' complement is an error if it belongs to more than one clusters. This approach was also used in the Manicure document processing system [9]. Another work of Taghva *et al.* [10] generated candidates by static/dynamic mappings and scored them using Bayesian function on frequencies of collocations and character ngrams. In addition, a heuristic was designed to create candidates for words containing unrecognized characters. The ranked list of candidates was then suggested to users. These approaches ignored POS contextual information.

Regarding the top competition approaches, there is a wide range of applied techniques. The best performing detector in the competition (*WFST-PostOCR*) [3] compiled probabilistic character error models into weighted finite-state edit transducers (WFST). Bigram language models and the lattice of candidates generated by the error model were used to decide the best token sequence. This approach disregarded some important contextual information, such as POS features and skip-grams.

Along with the development of machine translation techniques, some approaches considered OCR post-processing as machine translation (MT), which translates OCR text into the correct one in the same language. Two approaches (CLAM, Char-SMT/NMT) applied MT techniques at character level to detect and correct OCR errors. Another approach MMDT [11] combined many modules for candidate suggestion. Then, the decision module of MT technique was used to rank candidates.

In the 2-pass RNN competition approach [3], erroneous tokens were detected based on two recurrent neural network (RNN) models. Features of character level model were used as the input of the next model at word level.

Most of the prior approaches search for the best word candidates for each OCR token position. If the alternatives differ from the OCR token, then the token is erroneous and the best alternative is suggested to correct this error. The advantage of these approaches is to detect and correct errors at the same time. However, the performance of the detection task depends on that of a more difficult task - the correction.

In contrast to the above-discussed approaches, ours mainly focuses on detecting incorrect tokens. We exploit features at character level as well as word level and suggest two novel ones (the adapted version of the peculiar index and a frequency of an OCR token in its candidate generation sets) to classify the OCR token into incorrect/correct class. An OCR token needs to prove to be a valid word via feature values computed from its plausible candidate set.

III. Post-OCR error detection using binary classification

Our approach identifies whether an OCR token is correct/incorrect via binary classification. Feature values of each OCR token are computed relying on its candidate set. Therefore, our approach has one more step - candidate generation before typical steps of statistical approaches. In the following sections, we discuss each step in detail.

A. Candidate generation

In this section, we focus on generating possible candidates for each token position in OCR documents.

In order to produce candidates, we utilize information related to an OCR token at character (character error model) and word level. We consider character level as important as word level, therefore the same number of top candidates ($k = 5$) are used for each level.

At character level, we implement the approach similar to the one described in [12] to suggest candidates for each OCR token. In that work, candidates are generated from a character candidate graph and are ranked by the probabilistic character error model. For example, an OCR token "comes" can have its candidate set {"comes", "cones", "comas"}.

At word level, we make a use of local context to generate candidates. Word trigrams related to an OCR token are

considered. Let us assume the phrase $w_{-2}\ w_{-1}\ w\ w_{+1}\ w_{+2}$. The three trigrams involving the OCR token w are $w_{-2}\ w_{-1}\ w$ (denoted as *left-trigram*), $w_{-1}\ w\ w_{+1}$ (denoted as *middle-trigram*), and $w\ w_{+1}\ w_{+2}$ (denoted as *right-trigram*).

In case of the *left-trigram*, we keep w_{-2}, w_{-1} and select candidate w_0 for replacing the OCR token w. Then, we choose top k candidates based on the trigram frequency w_{-2}, w_{-1}, w_0. Similarly with the *middle-trigram* and *right-trigram*, top k possible candidates are chosen for the OCR token position.

For example, with the OCR token "the" in the phrase "his friend comes from the north we.t", its three related trigrams include "comes from the", "from the north", "the north we.t". Regarding the *left-trigram* "comes from the", we keep "comes from" and select candidates for the OCR token "the". Three trigrams with the highest frequency are "comes from the", "comes from a", "comes from an". Thus, top *k* candidates {"the", "a", "an"} of the *left-trigram* are chosen for the token position.

Similarly, we get word candidates for the OCR token position from the *middle-trigram* and *right-trigram*, which are, {"the", "up", "a"} and the empty set, respectively.

B. Feature extraction

Several features are extracted at character and word level. They can be divided into four groups: character n-gram frequency, word n-gram frequency, part-of-speech, and the frequency of OCR token in its candidate generation sets. It should be noted that due to shared characteristics between our datasets and the Corpus of Historical American English (COHA) [13], we use frequencies of n-grams and parts-of-speech (POS) of this largest corpus of historical English text. The CLAWS tagset is applied for all POS tags of this corpus.

1) Character n-gram frequency: The index of peculiarity (or the peculiar index) of a token is recommended for detecting OCR errors [1], therefore we consider the peculiar index as a classifier feature. In our work, we reuse the formula of computing the peculiar index on frequency statistics of the historical corpus COHA.

The key idea of the peculiar index is that if strings contain non-existent or very infrequent n-grams (like "jtg" or "bkm"), they are detected as potential erroneous tokens. The peculiar index of a token is the root-mean-square of the indices of its trigrams.

$$score(w) = \sqrt{\frac{\sum_{x \in trilist} index(x)^2}{n}} \quad (1)$$

where $trilist$ is the list of trigrams of the OCR token w, n is the size of $trilist$, and $index(x)$ is the index of trigram x which is computed as follows:

$$index(x) = \frac{\sum_{y \in bilist} log(freq(y))}{2} - log(freq(x)) \quad (2)$$

where $bilist$ is the list of bigrams of trigram x.

For example, we reuse the OCR phrase "his friend comes from the south we.t" with the OCR token "we.t" and its two trigrams {"we.", "e.t"}, the peculiar index of this token is:

$$score(\text{“}we.t\text{”}) = \sqrt{\frac{index(\text{“}we.\text{”})^2 + index(\text{“}e.t\text{”})^2}{2}}$$

$$index(\text{“}we.\text{”}) = \frac{log(freq(\text{“}we\text{”})) + log(freq(\text{“}e.\text{”}))}{2} - log(freq(\text{“}we.\text{”}))$$

Tokens with a higher index of peculiarity tend to be incorrect tokens [2]. However, it is unfair to compare the index of the peculiarity of long tokens and that of short tokens. Our analysis on the training part of our datasets suggests that long tokens tend to have a higher index of peculiarity. Therefore, we adapt the peculiar index to token length. In particular, for each dataset, we group the indices according to the length of their corresponding tokens; for each group, the peculiar index is normalized by the highest peculiar index of that group. The adapted index of peculiarity (denoted as *pe-index*) and the corresponding token length (denoted as *tok-len*) are used as classifier features.

By catching frequent character n-grams, the character level features have potential to correctly recognize out-of-vocabulary (OOV) words, which are often considered as errors because they are not present in the dictionary.

2) Word n-gram frequency: At word level, several features are extracted from contiguous ngrams to skip-ngrams.

Word frequency: The frequency of the OCR token w is normalized by the maximum frequency of its candidate set *C*, and is utilized as one feature value (denoted as *word-freq*).

$$score(w) = \frac{freq(w)}{\max_{c_i \in C}(freq(c_i))} \quad (3)$$

For example, for the OCR token "comes", its candidates with corresponding frequencies {"comes": 300, "cones": 100, "comas": 200}, the feature score of "comes" is calculated as follows:

$$score(\text{“}comes\text{”}) = \frac{300}{max(300, 100, 200)} = 1$$

Bigram frequencies of the OCR token and its neighbours: The bigram frequency of the OCR token and its previous token is normalized by the maximum bigram frequency of the OCR token's candidates and its previous token's candidates, and then is used as one feature (denoted as *pre-bi-freq*).

$$score(w) = \frac{\max_{i \in C_{-1}} freq(i, w)}{\max_{i \in C_{-1}, j \in C} freq(i, j)} \quad (4)$$

where C, C_{-1} are the candidate set of the OCR token, and that of its previous token.

For example, there are the OCR token "the" in the phrase "from the north", the candidate set C_{-1} {"from", "front"}, and the candidate set C {"the", "he", "she"}. The occurrence frequencies between "from", "front" and each candidate of

set C are {100, 2, 1}, {105, 2, 3}. The feature score of the OCR token "the" is computed as follows:

$$score(w) = \frac{\max(100, 105)}{\max(100, 105, 2, 2, 1, 3)} = 1$$

Similarly, the bigram frequency of the OCR token and its next word is normalized by the maximum bigram frequency of the OCR token's candidates and its next token's candidates, and is used as a feature (denoted as *next-bi-freq*).

$$score(w) = \frac{\max_{j \in C_{+1}} freq(w, j)}{\max_{i \in C, j \in C_{+1}} freq(i, j)} \quad (5)$$

where C, C_{+1} are the candidate set of the OCR token, and that of its next token, respectively.

In addition, the product of the previous bigram frequency and next bigram frequency (denoted as *prod-bi-freq*) is also considered as one feature.

Skip-gram frequencies: In order to reduce the data sparsity problem, skip-grams [14] are examined beside contiguous n-grams. Since COHA provides only 4-gram word frequencies, we focus on 2-skip-bigrams of an OCR token and its neighbors.

Frequencies of 2-skip-bigrams of the OCR token and three left/right words are calculated. The sum of these concurrent frequencies is normalized by the number of OCR token frequencies (*i.e.* divided by six in our work), and is then used as the skip-gram feature value. In particular, for an OCR token w in the phrase $w_{-3}\ w_{-2}\ w_{-1}\ w\ w_{+1}\ w_{+2}\ w_{+3}$, the score of skip-gram feature is calculated as follows:

$$score(w) = \frac{\sum_{w_i \in L_1} freq(w_i, w) + \sum_{w_j \in L_2} freq(w, w_j)}{6 * freq(w)} \quad (6)$$

where the list L_1 is $\{w_{-3}, w_{-2}, w_{-1}\}$, the list L_2 is $\{w_{+1}, w_{+2}, w_{+3}\}$.

For example, for the OCR token w "from" in the phrase "his friend comes from the north we.t", the skip-gram feature is computed as follows:

$$\begin{aligned} sumfreq = {} & freq(\text{“}his\text{”}, w) + freq(\text{“}friend\text{”}, w) \\ & + freq(\text{“}comes\text{”}, w) + freq(w, \text{“}the\text{”}) \\ & + freq(w, \text{“}north\text{”}) + freq(w, \text{“}we.t\text{”}) \end{aligned}$$

$$score(w) = \frac{sumfreq}{6 * freq(w)}$$

Split-word feature: While *run-on* errors are easily found by *non-word error detection* techniques, *incorrect split* errors are more challenging to identify. In order to deal with incorrect split errors (e.g. "made" vs. "ma lie"), we generate a split-word candidate list C_s from the OCR token w and its next token. For each candidate, we compute a score based on word frequency and bigram frequency, then use the highest score as a classifier feature (denoted as *split-word*), as shown in Eq. 7.

$$score = \max_{c_i \in C_s} (\alpha * x(c_i, w) + (1 - \alpha) * y(c_i, w)) \quad (7)$$

where w_{-1} is the previous word, $x(c_i, w)$ is 1 if $freq(c_i) > freq(w)$ else 0, $y(c_i, w)$ is 1 if $freq(w_{-1}, c_i) > freq(w_{-1}, w)$ else 0, α is the contribution rate between word frequency and bigram frequency, which was selected by keeping all other parameters same and trying different values between 0 and 1 with a step of 0.1. Our experiments showed that $\alpha = 0.5$ is the best level.

Consider, for example, the OCR phrase "they main tain good relations", two adjacent OCR tokens "main tain" and the previous token "they". The split-word candidate set C_s of "main tain" is {"maintain", "maintan", "niaintain"}. The OCR token "main" has the highest frequency in comparison to the candidates and only "maintain" has higher bigram frequecy than "main", hence the feature score is as below:

$$score = \max(0.5, 0, 0) = 0.5$$

3) Part-Of-Speech (POS) features: POS is considered as the general form of ngram feature. Our work utilize POS tag in length 3 as a classifier feature.

Let us assume that there are the OCR token w in the phrase $w_{-2}\ w_{-1}\ w\ w_{+1}\ w_{+2}$, and its candidate set C $\{c_1, c_2, c_3\}$. The tri-POSs related to the OCR token w are $pos_{-2}\ pos_{-1}\ pos$, $pos_{-1}\ pos\ pos_{+1}$, and $pos\ pos_{+1}\ pos_{+2}$ denoted as the *left-POS*/*mid-POS*/*right-POS*, respectively.

We first get a list of POS tags of each token $w_{-2}, w_{-1}, w, w_{+1}, w_{+2}$ from COHA corpus, denoted as L_{-2}, L_{-1}, L, L_{+1}, L_{+2}, respectively. OCR token's candidates c_1, c_2, c_3 also have the POS lists $L_{c_1}, L_{c_2}, L_{c_3}$, respectively.

As to the *left-POS*, all possible tri-POSs of the OCR token and its candidates are created by combining POS tags of L_{-2}, L_{-1} and each POS list of $\{L, L_{c_1}, L_{c_2}, L_{c_3}\}$.

The maximum frequency of the *left-POS* of OCR token is computed, then normalized by the maximum frequency of tri-POSs created from L_{-2}, L_{-1} and each POS list of $\{L_{c_1}, L_{c_2}, L_{c_3}\}$. The normalized *left-POS* frequency of OCR token is used as one feature.

$$score(w) = \frac{\max_{i \in L_{-2}, j \in L_{-1}, k \in L} freq(i, j, k)}{\max_{L_{c_i} \in L_c} \max_{i \in L_{-2}, j \in L_{-1}, k \in L_{c_i}} freq(i, j, k)} \quad (8)$$

with $L_c = \{L_{c_1}, L_{c_2}, L_{c_3}\}$.

We reuse the same OCR phrase to illustrate this feature, "his friend comes from the north we.t". The OCR token w is "comes", and its candidate set is {"comes", "cones", "comas"}. The POS lists of two previous words "his","friend" are {*appge, ppge*}, {*nn1, np1*}, respectively. The POS lists of "comes", "cones", "comas" are {*vvz*}, {*nn2*}, {*nn2*}, respectively. The possible *left-POS* of the OCR token "comes" include {*appge nn1 vvz*, *ppge nn1 vvz*, *appge np1 vvz*, *ppge np1 vvz*} and their maximum frequency is 10625. Similarly, the maximum frequencies of the *left-POS* of the candidates "comes", "cones", "comas" are 10625, 16425, 16425. Consequently, the feature score is: $score(\text{“}comes\text{”}) = 10625/16425 = 0.65$.

Similarly, the normalized *mid-POS*, *right-POS* frequencies of OCR token are computed. In total, there are four features scores extracted from POS, including *left-POS*, *mid-POS*, *right-POS*, and their product (denoted as *prod-POS*).

4) OCR token frequency in candidate generation sets: As mentioned in Sec III-A, there are four sources to generate error candidates: character error model, local word context (*left-trigram*, *middle-trigram*, and *right-trigram*). The number of appearances of the OCR token in the candidate sets is normalized by the number of candidate sets, then it is used as a feature (denoted as *tok-freq*).

While other features are built from individual words or word context, the *tok-freq* feature is designed from both of them. Therefore, we think that this combined feature can deal with *non-word* as well as *real-word* errors.

For example, the OCR token "the" in the phrase "his friend comes from the north we.t" has the noisy channel candidate set {"the", "she", "he"}. The local context candidate sets include the *left-trigram* set {"the", "a", "an"}, the *middle-trigram* set {"the", "up", "a"}, and the empty *right-trigram* set. As a result, the feature score is $score(\text{“}the\text{”}) = 3/4$.

C. Error classification

If the OCR token is an error, then its feature vector is labeled as 1, otherwise 0. Gradient Tree Boosting is one of the best performing classifiers [15], therefore we use it to train and classify OCR errors. In our experiments, we use the scikit-learn library [16] with the maximum of 5 nodes in the tree, 800 boosting stages, and default parameters.

IV. Experiments

A. Evaluation metrics

Our results are evaluated by the same metrics (Precision, Recall, F-measure) and the same evaluation tool as the ones used in the ICDAR 2017 competition[1].

B. Results

We compare our proposed approach with the six top performing approaches of the competition (CLAM, Char-SMT/NMT, EFP, MMDT, WFST-PostOCR, 2-pass RNN), which were presented in Sec. II. The performance shown in Table I indicates that our method outperforms the highest performing approaches in the competition, with 6% and 2% greater F-measure than the state-of-the-art (WFST-PostOCR) on Monograph and Periodical, respectively.

Furthermore, the performance on different error types (*non-word* vs. *real-word*) is clarified. In our opinion, due to the sparsity problem, our context features are not really effective in detecting *real-word* errors. In fact, despite applying possible features at word level (from ngram, skip-gram to part-of-speech) our approach enables to identify 43% of them on Monograph and 49% on Periodical.

[1]https://sites.google.com/view/icdar2017-postcorrectionocr/

Table I
Experimental results of OCR error detection task in English datasets of the competition

	Monograph			Periodical		
Approaches	P	R	F	P	R	F
CLAM	94%	52%	67%	-	-	-
Char-SMT/NMT	98%	51%	67%	89%	50%	64%
EFP	63%	77%	69%	53%	55%	54%
MMDT	83%	55%	66%	70%	32%	44%
WFST-PostOCR	67%	82%	73%	68%	68%	68%
2-pass RNN	58%	77%	66%	64%	68%	66%
Proposed approach	82%	76%	**79%**	81%	61%	**70%**

In case of *non-word* errors, our approach correctly detects the majority of *non-word* errors (96% of them on Monograph and 85% on Periodical). Most of the unidentified *non-word* errors are erroneous tokens related to numbers. In addition, we also take out-of-vocabulary (OOV) words into account. On average, 46% of OOV words in our datasets are correctly recognized.

C. Feature analysis

In total, our approach relies on 13 features to classify OCR errors. Three features (*pe-index*, *tok-len*, *word-freq*) built from individual words mainly focus on detecting *non-word* errors. The features created from word context (bigram frequencies, skip-grams, POS tags) concentrate on identifying *real-word* errors. Regarding the combined feature *tok-freq*, it has potential to deal with both of *non-word* and *real-word* errors. The remaining feature *split-word* is designed to handle incorrect-split errors.

Each feature contributes differently to predicting the target. The more frequently a feature is used in the split points of a decision tree, the more important that feature is. In ensemble classifier like Gradient Tree Boosting, the feature importance is the average one of each tree [16]. As our approach detects more *non-word* errors than *real-word* ones, we believe that the features based on individual words (*pe-index*, *tok-len*, *word-freq*) and the combined feature *tok-freq* are more important than the features relying on word context. The relative importance of the features shown in Fig. 1 supports our assumption. In particular, *tok-freq* is the most important one (36.8% on Monograph and 52.3% on Periodical) and *pe-index* is the third one (15% on Monograph and 9.4% on Periodical).

The importance of our novel features are evaluated separately in Table II. In overall, *pe-index* and *tok-freq* help to increase recall on Monograph and precision on Periodical. This trend can be partly explained by characteristics of these features and different rates of non-word/real-word errors of the two datasets. In fact, *pe-index* and *tok-freq* mainly detect non-word errors, and the rates of non-word/real-word in Monograph are higher than in Periodical (1.5 vs. 0.5).

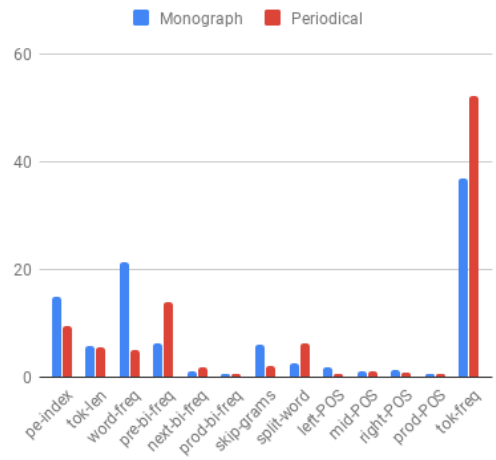

Figure 1. The relative importance of the features

Table II
PERFORMANCE OF OUR MODELS WITHOUT SOME NOVEL FEATURES

Models	Monograph			Periodical		
	P	R	F	P	R	F
11 features	87%	31%	46%	76%	59%	66%
11 features+*pe-index*	91%	48%	63%	80%	61%	69%
11 features+*tok-freq*	82%	74%	78%	80%	58%	67%
All features	82%	76%	**79%**	81%	61%	**70%**

V. CONCLUSIONS

This work examined a novel OCR error detection approach, which has better performance than the state of the art. The main idea of our method is to check whether an OCR token is a correct one using feature values computed from its plausible candidate set. Moreover, two novel features (*pe-index*, *tok-freq*) are found to play important roles in the detection of incorrect OCR tokens. Our future work will concentrate on correcting OCR errors with the detected error list. In addition, we will reuse these classifier features to suggest the best candidate correction of each OCR error.

ACKNOWLEDGMENT

This work has been supported by the European Union's Horizon 2020 research and innovation programme under grant 770299 (NewsEye).

REFERENCES

[1] K. Kukich, "Techniques for automatically correcting words in text," *Acm Computing Surveys (CSUR)*, vol. 24, no. 4, pp. 377–439, 1992.

[2] R. Morris and L. L. Cherry, "Computer detection of typographical errors," *IEEE Transactions on Professional Communication*, pp. 54–56, 1975.

[3] G. Chiron, A. Doucet, M. Coustaty, and J.-P. Moreux, "Icdar2017 competition on post-ocr text correction," in *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, vol. 1. IEEE, 2017, pp. 1423–1428.

[4] E. Zamora, J. J. Pollock, and A. Zamora, "The use of trigram analysis for spelling error detection," *Information Processing & Management*, pp. 305–316, 1981.

[5] A. R. Golding and Y. Schabes, "Combining trigram-based and feature-based methods for context-sensitive spelling correction," in *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, 1996, pp. 71–78.

[6] J. Mei, A. Islam, Y. Wu, A. Moh'd, and E. E. Milios, "Statistical learning for ocr text correction," *arXiv preprint arXiv:1611.06950*, 2016.

[7] G. Khirbat, "Ocr post-processing text correction using simulated annealing (opteca)," in *Proceedings of the Australasian Language Technology Association Workshop 2017*, 2017, pp. 119–123.

[8] K. Taghva, J. Borsack, B. Bullard, and A. Condit, "Post-editing through approximation and global correction."

[9] K. Taghva, A. Condit, J. Borsack, J. Kilburg, C. Wu, and J. Gilbreth, "Manicure document processing system," in *Document Recognition V*, vol. 3305. International Society for Optics and Photonics, 1998, pp. 179–185.

[10] K. Taghva and E. Stofsky, "Ocrspell: an interactive spelling correction system for ocr errors in text," *International Journal on Document Analysis and Recognition*, vol. 3, no. 3, pp. 125–137, 2001.

[11] S. Schulz and J. Kuhn, "Multi-modular domain-tailored OCR post-correction," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 2716–2726.

[12] T.-T.-H. Nguyen, M. Coustaty, A. Doucet, A. Jatowt, and N.-V. Nguyen, "Adaptive edit-distance and regression approach for post-ocr text correction," in *International Conference on Asian Digital Libraries*, 2018, pp. 278–289.

[13] M. Davies, "The corpus of contemporary american english as the first reliable monitor corpus of english," *Literary and linguistic computing*, pp. 447–464, 2010.

[14] A. Reyes, P. Rosso, and T. Veale, "A multidimensional approach for detecting irony in twitter," *Language resources and evaluation*, pp. 239–268, 2013.

[15] J. Wainer, "Comparison of 14 different families of classification algorithms on 115 binary datasets," *arXiv preprint arXiv:1606.00930*, 2016.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, pp. 2825–2830, 2011.