Received 13 March 2023, accepted 6 May 2023, date of publication 6 June 2023, date of current version 15 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3283340

RESEARCH ARTICLE

An Efficient Unsupervised Approach for OCR Error Correction of Vietnamese OCR Text

QUOC-DUNG NGUYEN[1], NGUYET-MINH PHAN[2], PAVEL KRÖMER[3], (Senior Member, IEEE), AND DUC-ANH LE[4,5], (Member, IEEE)

[1]Faculty of Mechanical-Electrical and Computer Engineering, School of Technology, Van Lang University, Ho Chi Minh City, Vietnam
[2]Faculty of Information Technology, Saigon University, Chi Minh City, Vietnam
[3]Department of Computer Science, VSB–Technical University of Ostrava, 708 00 Ostrava, Czech Republic
[4]The Institute of Statistical Mathematics, Tokyo 101-8430, Japan
[5]NTT Hi-Tech Institute, Nguyen Tat Thanh University, Ho Chi Minh City, Vietnam

Corresponding author: Pavel Krömer (pavel.kromer@vsb.cz)

This work was supported in part by the Van Lang University, Vietnam; in part by the European Regional Development Fund within the Operational Program Research, Development and Education in Projects, "A Research Platform focused on Industry 4.0 and Robotics in Ostrava Agglomeration" under Grant CZ.02.1.01/0.0/0.0/17_049/0008425; and in part by the VSB–TU Ostrava, Czech Republic, through the Student Grant System (SGS) under Grant SP2023/12 and Grant SP2023/76.

ABSTRACT Different types of OCR errors often occur in OCR texts due to the low quality of scanned document images or limitations in OCR software. In this paper, we propose a novel unsupervised approach for OCR error correction. Correction candidates for OCR errors are generated and explored in their neighborhoods using correction character edits controlled by an adapted hill-climbing algorithm. Correction characters are extracted from only original ground truth texts, which do not depend on OCR texts in training data. A weighted objective function used to score and rank correction candidates is heuristically tested to find optimal weight combinations. The proposed model is evaluated on an OCR text dataset originating from the Vietnamese handwritten database in the ICFHR 2018 Vietnamese online handwritten text recognition competition. The proposed model is also verified concerning its stability and complexity. The experimental results show that our model achieves competitive performance compared to the other models in the ICFHR 2018 competition.

INDEX TERMS OCR, character edit, error correction, attention-based encoder-decoder, hill climbing.

I. INTRODUCTION

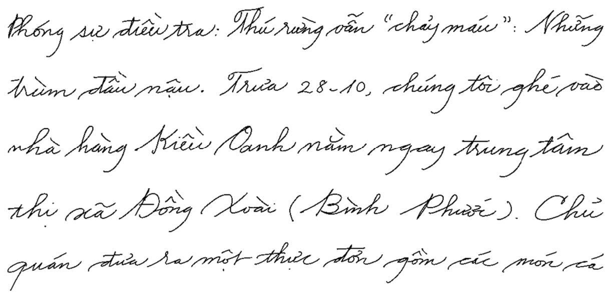
Optical Character Recognition (OCR) is a software tool for digitizing scanned document images of typed, handwritten, or printed texts into computer-readable forms using various image processing and pattern recognition techniques [1], [2], [3], [4], [5]. The computer-readable texts are known as OCR texts. The low quality of scanned documents and the limitations of OCR software often lead to different types of errors in OCR texts. These OCR errors negatively impact the downstream processes and applications which make use of OCR texts, such as document retrieval, search, indexing, or natural language processing tasks [6], [7], [8], [9], [10], [11], [12], [13], [14].

The associate editor coordinating the review of this manuscript and approving it for publication was Diego Oliva.

An example of Vietnamese OCR text is shown in Figure 1. Different types of errors are encountered, for example, misspelled errors due to character deletion/insertion/substitution operations (e.g.,"trùm" is misrecognized as "tràm", "Oanh" is wrongly recognized as "Qnh"); split word errors (e.g., "Phước" is split into "A hước"); case-sensitive errors (e.g., "Kiều" → "tiều"); punctuation errors (e.g., "tra:" → "tra.", "đầu" → "đầu,"), etc.

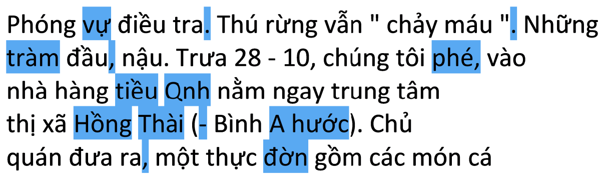
OCR post-processing aims to automatically detect and correct misspellings and linguistic errors in OCR texts. OCR post-processing approaches have been proposed using various methods such as corpus-based language models [15], [16], machine learning [17], [18], [19], evolutionary algorithms [20], [21], and statistical and neural machine translation [22], [23], [24], [25], [26].

The above approaches have exploited both linguistic features and OCR error characteristics for OCR error correction.



Phóng sự điều tra: Thú rừng vẫn "chảy máu": Những trùm đầu nậu. Trưa 28-10, chúng tôi ghé vào nhà hàng Kiều Oanh nằm ngay trung tâm thị xã Đồng Xoài (Bình Phước). Chủ quán đưa ra một thực đơn gồm các món cá

(a) Scanned text



Phóng vự điều tra. Thú rừng vẫn " chảy máu ". Những
tràm đầu, nậu. Trưa 28 - 10, chúng tôi phé, vào
nhà hàng tiều Qnh nằm ngay trung tâm
thị xã Hồng Thài (- Bình A hước). Chủ
quán đưa ra, một thực đờn gồm các món cá

(b) OCR text

Figure 1. **Example of a Vietnamese OCR text**[1].

While linguistic features can be characterized by syntactical and semantic features in a reference language, OCR error characteristics present typical features of OCR errors caused by the OCR process. Furthermore, most of them are supervised models since they use training data, including both Ground Truth (GT) texts [27] and OCR texts that are often aligned in advance at the character level or word level.

Candidate word generation by character edit operations is an important method that has been employed in many spelling correction and OCR error correction approaches [17], [19], [21], [28], [29], [30]. These approaches employed an error model using character edits learned from the datasets of spelling errors or OCR errors. The error model captures the error characteristics originating from the error sources, e.g., OCR error characteristics caused by the OCR process, or spelling error characteristics that come from mistyping made by users. In particular, Jurafsky and Martin [29] presented a noisy channel model to estimate the conditional probability of potential candidates for a given spelling error that is caused by unexpected character edits (e.g., mistyping), and the most probable candidate is chosen to replace the spelling error. In [21] and [28], we propose a method to generate correction candidates for OCR errors by using character edits that are learned and constructed from training data of aligned GT texts and OCR texts. Character edits can belong to edit operations of character insertion, deletion, substitution, and transposition. Character edits are applied to an OCR error to transform it into a correction candidate. However, character edits possibly generate unexpected correction candidates when evaluated on test data. The reason is that the constructed character edits from the training data might not cover new character edits that only occur in the test data.

As far as we know, no OCR post-processing approaches based on evolutionary or optimization-based algorithms have been proposed for Vietnamese OCR texts, not like for other languages such as English [20], [21], [31]. In this paper, we propose a novel unsupervised OCR error correction model based on an adapted Hill Climbing (HC) algorithm without knowledge of OCR error characteristics. The HC algorithm directs the exploration and selection of correction candidates in their neighborhoods using random correction character edits extracted from original GT texts. In this approach, we have no need of preprocessing training data by aligning GT text and OCR text at character level or word level as in the supervised approaches. The proposed model is evaluated on a Vietnamese OCR text dataset originating from the VNOnDB database used in the ICFHR 2018 Vietnamese online handwritten text recognition competition [32].

In summary, this study makes the following five contributions:

- We introduce a novel unsupervised approach for OCR error correction using random correction pattern edits through evolution loops controlled by the hill climbing algorithm.
- We propose a method to construct a correction pattern dictionary from ground truth texts, without depending on OCR texts in the training dataset. The same method can be applied to a larger text corpus to obtain a larger correction pattern dictionary.
- We employ a weighted objective function based on suggested and adapted linguistic features for scoring and selecting correction candidates.
- With efficient settings of the model parameters, our proposed model can perform with high quality in candidate generation and error correction, which can be tested on a normal personal computer. Our model outperforms most of the baseline models on the same evaluation dataset of the ICFHR 2018 competition.
- Our model can be employed as a tool for OCR error correction in different domains and languages.

The rest of the paper is structured as follows. Section II provides the overview of the recent OCR post-processing approaches as well as the description, applications, and adoption of the hill climbing algorithm in the OCR error correction problem. In Section III, the Vietnamese OCR text dataset is described, and the proposed OCR error correction model based on an adapted HC algorithm is presented in detail. The experimental results of the proposed model are discussed and compared to the other baseline models in Section IV. In addition, our proposed model is verified and optimized with different feature weight combinations. Our model is further examined regarding its stability and complexity. Finally, Section V gives conclusions and future work.

[1]Extracted from the Vietnamese OCR evaluation dataset described in Section III-A.

II. RELATED WORKS

A. OCR POST-PROCESSING APPROACHES

In general, there are three possible ways to improve the quality of OCR texts: modifying input images, optimizing the OCR system, and post-processing OCR output texts [33]. The third way is preferable as it does not rely on any specific parameters of the OCR system. It aims to automatically detect and correct different types of errors in OCR texts.

Automatic OCR post-processing approaches can be categorized into two main groups: corpus-based type and hybrid type. In the corpus-based approaches [15], [16], *n*-gram dictionaries are constructed from large external corpora and resources and used for OCR error detection and correction. However, the performance of this approach is limited due to the coverage of the constructed dictionaries. This is because such external resources only cover specific genres of documents and are available for a period of time. The second type of approach is often a combination of different models and techniques among language models based on *n*-grams [29] and noisy channel/error models [28], [34], machine learning [17], [18], [19], statistical and neural machine translation [22], [23], [24], [25], [26], and evolutionary and optimization algorithms [20], [21].

Several typical examples of hybrid approaches are summarized in the following. Kissos and Dershowitz [17] employ the OCR error model with confusion matrix and word *n*-gram models for correction candidate generation. They train the regression model on six linguistic features, including confusion weight with a single character edit, word frequency, backward and forward bigram frequency, term frequency in OCR text, and word confidence for candidate ranking and selection. Similar approaches can be found in Mei et al. [18] and Nguyen et al. [19]. Khirbat [20] utilizes a support vector machine trained using a given training dataset for error detection and employs a confidence-based mechanism using the simulated annealing algorithm to obtain an optimal candidate from a pool of correction candidates. In our recent work [21], we have employed *n*-gram language models and a candidate generation model based on correction pattern edits and the self-organizing migrating algorithm [35].

Afli et al. [22], [23] apply a statistical machine translation (SMT) model to translate the OCR output text into the corrected text in the same language, following the work [36]. The multi-modular domain-tailored approach [24] consists of multiple modules for candidate suggestion, ranking, and selection. These modules include word-level suggestion modules (original words, spell checker, compounder, word splitter, text-internal vocabulary), sentence-level suggestion modules based on character-level and word-level SMT models, and decision module with the Moses decoder [37]. Amrhein and Clematide [25] propose an ensemble approach of character-based statistical and neural machine translation models (SMT/NMT), and such models are trained and used to generate the best candidates for OCR errors. Nguyen et al. [26] employ pre-trained bidirectional encoder representations from transformers models (BERT) at word level for OCR error detection and character-level NMT techniques using the OpenNMT toolkit [38] for error correction. In addition, Afli et al. [23] pointed out that the word-level machine translation has better performance than the character-level one. Furthermore, the character-level machine translation might generate non-word candidates that do not appear in the word dictionaries.

Regarding the Vietnamese language, only a few OCR post-processing approaches [28], [39], [40] have been proposed for Vietnamese OCR texts. In [28], we make use of the word-level *n*-gram models and the error model using correction character edits learned from aligned GT and OCR training texts to detect and correct OCR errors. We also propose a word-level NMT model based on the bidirectional long short-term memory (BiLSTM) network to correct OCR errors in our recent work [39]. Hoang and Aw [40] apply two correction models. The weighting-based model employs two linguistic features, syllable similarity and syllable frequency, and the contextual corrector uses language modeling based on a perplexity score. In the second model, they implement the depth-first traversal algorithm to examine all candidate combinations, leading to a high computational cost due to the exploding number of combinations for the high number of nodes (syllables). In addition, they have used their own dataset for the model evaluation, causing inconvenience in comparison with other approaches.

B. HILL CLIMBING ALGORITHM

Optimization algorithms often employ local and global heuristic search strategies to find solutions in search space for solving optimization problems. Hill climbing is an optimization algorithm that belongs to the local search family [30]. It is often used to produce approximate solutions for optimization problems when the amount of time available to perform a search is restricted, such as with real-time systems. The relative simplicity of the HC algorithm makes it a popular first choice among optimization algorithms. The HC algorithm is also categorized into a group of point-based algorithms since it deals with a single candidate at each step. The HC algorithm has found its applications in the optimization domain such as the traveling salesman problem [41], binary search [42], planning problem [43], and in other domains such as text decoding [30], neural network training [44], and game playing [45].

Hill climbing is an iterative algorithm that starts at a random location (a solution to a problem), *h*, then takes a step to a neighboring location by making an incremental change to *h*. If that location is higher (better solution with higher fitness value), the algorithm continues to hill-climb from there; otherwise, it moves to another neighbor of *h*. The algorithm repeats until a certain number of steps are reached or no further improvements can be made. Random restarts can be added to the algorithm to prevent it from getting stuck on

Algorithm 1 The Hill-Climbing Algorithm

```
1  function hillclimb($h, f, neighbors, steps$);
     Input  : - Random starting location $h$
              - Fitness function $f$
              - Neighborhood function that generates
              the neighbors of a location $neighbors$
              - Maximum number of steps to take
              before the algorithm is terminated $steps$
     Output: Optimal location $h_{max}$
2    $h_{max} = h$;
3    $f_{max} = f(h)$;
4    $neighborhood = neighbors(h)$;
5    for $i$ from 1 to $steps$ do
6        for each $h$ in $neighborhood$ do
7            if $f(h) > f_{max}$ then
8                $h_{max} = h$;
9                $f_{max} = f(h)$;
10               $neighborhood = neighbors(h)$;
11               break ;  /* terminate the for
                  loop of neighborhood and
                  continue next step */
12           end
13       end
14   end
15   return $h_{max}$;
```

a local hilltop or wandering around a flat plain. The original hill climbing algorithm is described in Algorithm 1.

The central part of OCR error correction approaches is candidate generation. The aim of candidate generation is to find as many correction candidates as possible in the search space, and it has a better chance to find an optimal candidate with the highest fitness value among all possible correction candidates. Finding the most probable candidate among all possible correction candidates of an OCR error can be considered as a search problem.

However, there is a trade-off between searching such many possible correction candidates in the search space and the computational cost of the search. Apparently, searching all possible correction candidates for each error syllable will increase computational time. In addition, a number of candidates in the search space are possibly not good candidates in terms of fitness value. Therefore, it is necessary to have a way of finding optimal candidates without searching through every possible candidate. That suggests using an optimization algorithm.

We observe that the HC algorithm has properties that can be adapted and applied to the OCR error correction problem. Firstly, candidate solutions generated through the climbing steps can be mapped to correction candidates of an error syllable s_e. Secondly, the original HC algorithm starts at a random location and then hill-climbs to a higher location in the neighborhood of the starting location. In the adapted algorithm, we first randomly select character positions of the error syllable s_e at the beginning of the algorithm, and then in each step, we apply character edits at the randomly selected character positions of s_e to generate a candidate. This generated candidate is checked if it occurs in a vocabulary. If it does, it is selected as a correction candidate; otherwise, it is discarded. Thirdly, the fitness value of each solution can be employed as the weighted score of the linguistic features extracted from each candidate word and its context words. Finally, the algorithm can be restarted to improve the quality of generated candidates, the same idea as in the original algorithm. Furthermore, the randomness incorporated into the algorithm steps can improve the performance in correction candidate generation since it increases the diversity of generated candidates of s_e.

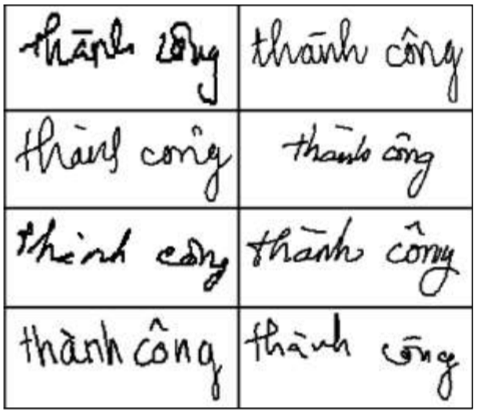


Figure 2. Examples of different writing styles of the same syllables.

III. DATASET AND PROPOSED MODEL

In this section, we describe the Vietnamese OCR text dataset used for model evaluation. Then, the proposed OCR post-processing model and its processing phases are presented in detail. In Vietnamese, each space-separated token (word unigram) is in monosyllabic form. Therefore, we call each token a syllable.

A. DATASET

The VNOnDB[2] benchmark database [4] was used in the Vietnamese online handwriting recognition competition[3] [32] to encourage the studies of Vietnamese handwriting recognition. The database provides unconstrained handwritten texts written by 200 Vietnamese writers in their own styles without any restriction (see Figure 2). It comprises 1,146 Vietnamese paragraphs, 7,296 lines, more than 480,000 strokes, and more than 380,000 characters (without the space character). Three datasets at paragraph, line, and word levels are VNOnDB-Paragraph, VNOnDB-Line, and VNOnDB-Word respectively, provided with the ink data and ground truth.

[2]http://tc11.cvc.uab.es/datasets/HANDS-VNOnDB2018_1, accessed 09 August 2022.

[3]https://sites.google.com/view/icfhr2018-vohtr-vnondb/home, accessed 09 August 2022.

Table 1. **The statistics of the VNOnDB database.**

Dataset	Training	Validation	Test
# Paragraphs	690	196	258
# Lines	4,433	1,229	1,634
# Syllables	66,991	18,640	25,115

Table 2. **Error rates of the VNOnDB-OCR dataset at the syllable level and character level with respect to the training/validation and test dataset types.**

Level	Dataset type	# Error count	Error rate (%)
Syllable	Training and validation	3,809	4.3
	Test	3,524	13.6
Character	Training and validation	6,505	1.7
	Test	5,245	4.7

The ground truth texts come from the VietTreeBank[4] corpus (VTB) [46]. Each dataset is divided into the training, validation, and test sets, and their statistics are given in Table 1.

In our experiments, the OCR texts are the outputs from an OCR engine based on the attention-based encoder-decoder model (AED) [5] with the DenseNet encoder and the attention-based long short-term memory decoder. The handwritten images converted from the corresponding online handwritten texts in the VNOnDB-Line dataset are inputted into the AED model to produce the OCR texts. In the VNOnDB-Line dataset, many text lines are not complete (without making a complete sentence), see Figure 3. The characters are not even straight on these text lines. Furthermore, in unconstrained Vietnamese handwritten text, the place, size, and shape of diacritic marks (DM) often vary due to different writers or even different writing times of the same writer. Such problems as distorted DMs, incomplete text lines, and free handwriting styles will create more challenges for the OCR process and result in different types of errors in generated OCR texts.

We call the collection of OCR texts generated from the AED model and their corresponding GT texts the **VNOnDB-OCR** dataset. There are 5,662 GT-OCR text line pairs from the training and validation sets (called the training dataset), and 1,634 text line pairs from the test set for model evaluation (called the test dataset).

Based on the same VNOnDB-OCR dataset, our study [47] provided the error rates at the syllable level and character level as shown in Table 2. We can see that the error rates in the test dataset (4.7% and 13.6% at the character level and syllable level, respectively) are much higher than those in the combined training and validation dataset (1.7% and 4.3%). The higher error rate in the test dataset would create difficulties for OCR post-processing approaches since new error character patterns could only occur in the test dataset.

B. PROPOSED MODEL

In this section, we present the proposed OCR post-processing model (see Figure 4). The model includes three processing phases: tokenization, error detection, and error correction. In the OCR error correction phase, correction candidates are discovered through the search process using random correction character edits directed by the HC algorithm. Candidates are ranked and selected among evolution loops using the objective function based on the different linguistic features. In this model, character edits are extracted from only the original GT texts in the training dataset, without using the OCR texts.

During these processing phases, we need to make use of the syllable n-grams for error detection, candidate search, and candidate ranking and selection (see the details in the following error detection and error correction sections). Three dictionaries of unigrams, bigrams, and trigrams along with their observed frequency counts are constructed from the VTB corpus. The n-gram dictionaries are further enriched with the GT training texts. The unigram dictionary is also called the syllable vocabulary. Similarly, for a specific language based on the Latin alphabet like English, French, German, etc., the same method can be applied to construct the n-gram dictionaries using a large external corpus and GT training data in this specific language.

[4]https://vlsp.hpda.vn/demo/?page=resources, accessed 17 August 2022.

Table 3. **The tokenized OCR text lines.**

ID	Example
1	Thành|công|ấy,|liệu|có|mấy|người|đạt|được? [a]
2	Sau|mỗi|mùa|thi|đại|học,|có|bao|“|sĩ|tử|”|buồn|rầu|khi|biết|mình|trở|thành|“|tử|sĩ|”.
3	diện|tích|trên|1|triệu|km2|(gấp|3|diện|tích|đất|liền|:|1|triệu|km2|/|330.000|km2).
4	“|Trang|trại|mà|đã|xuất|hiện|thì|làm|sao|rừng|yên!|”,|ông|Phạm|Văn|Án|-|chi|cục|trưởng|Kiểm|lâm|tỉnh|Lâm|Đồng|-|đã|từng|nói|vậy...
5	Tay|“|lái|hổ|”|nói|:|“|Nó|vừa|từ|chuyên|cơ|hạ|cánh|xuống|VN;|nặng|2,7|tạ|(khi|sống),|giá|500|triệu|đồng|”.
6	Thứ|trưởng|Bộ|Tài|nguyên|&|môi|trường|Đặng|Hùng|Võ|cho|biết|dự|thảo|nghị|định

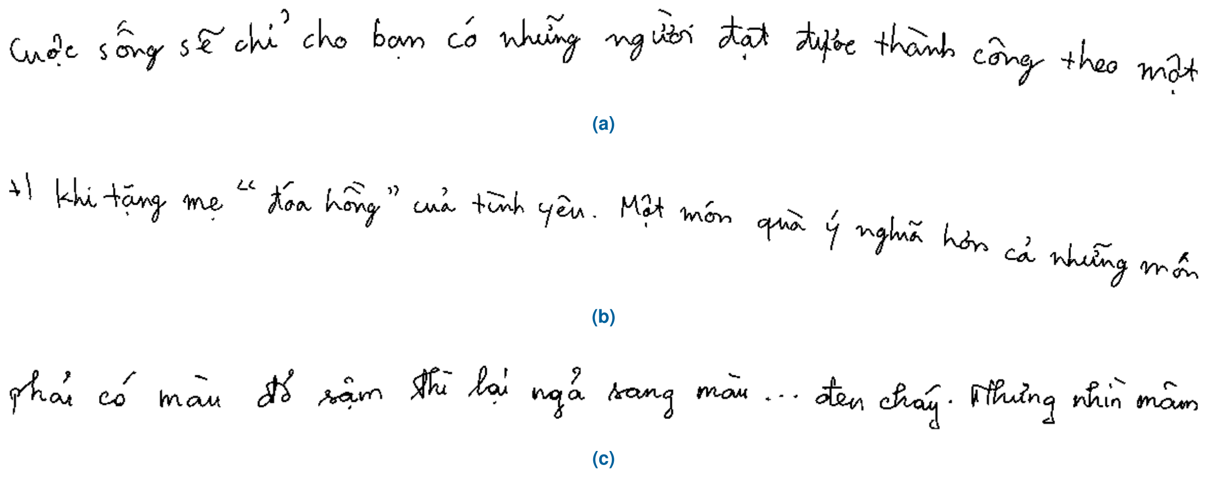
[a]The “|” character presents the space character.

1) TOKENIZATION

The OCR texts are tokenized into space-separated syllables with no restriction on punctuation. Several special characters such as “**-**”, “**"**”, “**/**”, “**:**”, “**&**” are also considered as syllables; thus, they are also separated on white space. Table 3 displays the OCR texts after tokenization. The tokenized OCR texts will be utilized in the error detection and correction phases later. Besides, the punctuation marks, occurring at the end of tokens such as “**.**”, “**,**”, “**;**”, “**?**”, “**!**”, “**)**”, and the opening parenthesis “**(**” occurring at the start of tokens, are removed before the error detection is processed. It is because these punctuation marks positioned at the start or end of n-grams were also detached in the constructed n-gram dictionaries.

2) ERROR DETECTION

Each OCR text in the test dataset is scanned through to identify non-syllable errors for the syllables that do not occur


Cuộc sống sẽ chỉ cho bạn có những người đạt được thành công theo một

(a)

+) khi tặng mẹ “ đóa hồng” của tình yêu. Một món quà ý nghĩa hơn cả những món

(b)

phải có màu đỏ sậm thì lại ngả sang màu ... đen cháy. Nhưng nhìn mâm

(c)

Figure 3. **Examples of incomplete source text lines.**

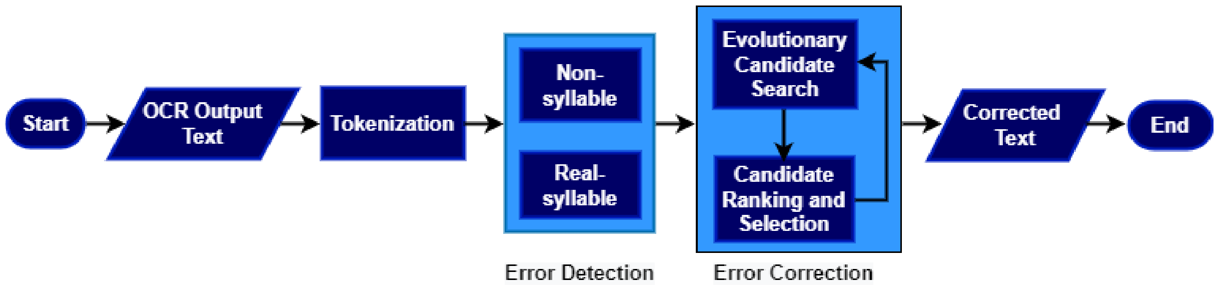


Figure 4. **The processing phases of the proposed OCR post-processing approach.**

in the syllable vocabulary. For every real syllable that occurs in the vocabulary, its bigrams and trigrams are examined. If none of them exist in the bigram and trigram dictionaries, the real syllable is considered erroneous. In particular, given a real syllable s_i, the left and right bigram contexts of s_i are $s_{i-1}s_i$ and s_is_{i+1}; while three trigram contexts of s_i are $s_{i-2}s_{i-1}s_i$, $s_{i-1}s_is_{i+1}$, and $s_is_{i+1}s_{i+2}$, where s_{i-2} and s_{i-1} are the preceding syllables of s_i; similarly, s_{i+1} and s_{i+2} are the following ones of s_i.

The different types of errors are observed in the error detection as shown in Table 4. The "→" symbol represents the left GT syllable (or phrase) that is incorrectly recognized as the right OCR syllable (or phrase). For a detailed description of the character-level and syllable-level errors and their possible causes, see our previous study [47]. Our proposed OCR post-processing model will focus on detecting and correcting character-level errors.

Table 4. **Vietnamese OCR error classification.**

Error Type		**Example**
Character level	Non-syllable	thu → thu**y** (Insertion)
		cã**i** → cã (Deletion)
		sự → **v**ự, **Oa**nh → **Q**nh (Substitution)
	Real-syllable	Trưng → Trư**ơ**ng (Insertion)
		ki**ê**m → kim (Deletion)
		c**u**ng → c**ứ**ng; **qu**à → **m**à (Substitution)
Syllable level	Inserted / deleted syllable	cũng|không|ai|cho|chúng|tôi|biết|phải|làm|gì → cũng|không|**có**|ai|cho|chúng|tôi|biết|phải|làm|gì (Insertion)
		kèm|theo|lở|núi|đổ|**ào**|xuống|trại → kèm|theo|lỡ|núi|đổ|xuống|trại (Deletion)
	Inserted / deleted / substituted punctuation	anh|muốn|chúng|tôi|mang|xuống|tận|nhà|hàng → anh|muốn|chúng|tôi|mang|xuống|tận**,**|nhà|hàng ("**,**"wrongly inserted in the OCR text)
		con|cá|tự|nhiên|trong|đầm**.** → con|cá|tự|nhiên|trong|đầm ("**.**"wrongly deleted from the OCR text)
		"|Hương|rừng|"|trong|thành|phố**!** → "|Hương|rừng|"|trong|thành|phố**.** ("**!**"in the GT text substituted by "**.**"in the OCR text)

3) ERROR CORRECTION

The error correction phase involves two processes: candidate search, and candidate ranking and selection. In the candidate search process, correction candidates are explored through evolution loops under the control of the adapted HC algorithm. Then, they are scored and selected using the objective function based on the adopted linguistic features in the candidate ranking and selection process.

a: CANDIDATE SEARCH

A character edit is a character mapping between an error syllable s_e and a correction syllable s_c, and can be defined

Algorithm 2 Correction Pattern Extraction

```
1  function corrpatterns($T_{GT}$);
      Input  : GT texts in the training dataset
               $T_{GT} = t_1, t_2, \ldots, t_N$
      Output: $D_{cor}$   /* A dictionary of (key, value)
               tuples, wherein key is a correction
               pattern, and value is its corresponding
               occurrence frequency */
2     initialize $D_{cor}$ as an empty dictionary;
3     for text t from $t_1$ to $t_N$ do
4         $S_{token} \leftarrow tokenize(t)$ ;  /* tokenize the
           text t */
5         for each token s in $S_{token}$ do
6             for i from 1 to $len(s) - 1$ do
7                 $D_{cor}[s[i]] = D_{cor}.get(s[i], 0) + 1$ ;
                   /* add one-character
                   pattern $s[i]$ into $D_{cor}$ */
8                 $D_{cor}[s[i]s[i+1]] =$
                   $D_{cor}.get(s[i]s[i+1], 0) + 1$ ;
                   /* add two-character
                   pattern $s[i]s[i+1]$ into $D_{cor}$
                   */
9             end
10            $D_{cor}[s[len(s)]] = D_{cor}.get(s[len(s)], 0) + 1$
               ;      /* last one-character
               pattern $s[len(s)]$ */
11        end
12    end
13    return $D_{cor}$;
```

as $e : s_{e_{i:i+m-1}} \rightarrow s_{c_{j:j+n-1}}$, where $s_{e_{i:i+m-1}}$ is the consecutive characters of length m in the error syllable s_e, and $s_{c_{j:j+n-1}}$ is the sequence of n characters in the correction syllable s_c. The $s_{e_{i:i+m-1}}$ characters (called error character pattern) are substituted by $s_{c_{j:j+n-1}}$ (called correction character pattern) to transform s_e into s_c. A character edit can belong to four edit operations: insertion, deletion, transposition, and substitution (transposition can be considered a special case of substitution). The error syllable s_e can be corrected by several edit operations. For example, the error syllable "nhưa" is corrected to "chưa" by one substitution operation "n"→"c"; "chủy" corrected to "khủng" has two substitutions "c"→"k" and "y"→"ng"; "A hước" can be corrected to "Phước" by one substitution "A"→"P" and one space deletion; "trực?" can be corrected to "trả nợ" with three operations including one substitution "ự"→"ả", one insertion "“"→" n" (including a space character), and one more substitution "c?"→"ợ". We observe these errors in the VNOnDB-OCR dataset originated from different-style writers.

According to our study [47] on the same dataset, the top mappings at both character level and syllable level are of one- or two-character length of edits. Hence, we employ character edits with length of one or two characters to generate correction candidates. In other words, both the error character pattern and the correction character pattern of a two-character edit will be restricted to a maximum length of two characters.

We use the GT texts in the training dataset to construct correction patterns of 1 or 2 characters long. The occurrence frequency of correction patterns in the GT text corpus is also recorded. A method to extract correction patterns from the GT text corpus is shown in Algorithm 2. It is assumed that the function *get*() returns the value of the specified key; if the key does not exist, it returns 0.

After the correction pattern dictionary D_{cor} is constructed, candidates for detected OCR errors are generated by substituting error character patterns with correction character patterns obtained from the D_{cor} dictionary. Candidates are discovered through the search process using random correction patterns directed by the adapted HC algorithm as presented below.

Given the error syllable s_e, d random positions along the characters of s_e are selected at the start of the adapted HC algorithm. The parameter d is equivalent to the number of edit operations (also known as edit distance) applied to s_e to generate candidates. The algorithm consists of two main processes: climbing and selection. The flow of the algorithm is described in Figure 5.

- The climbing process includes the steps in the following order:

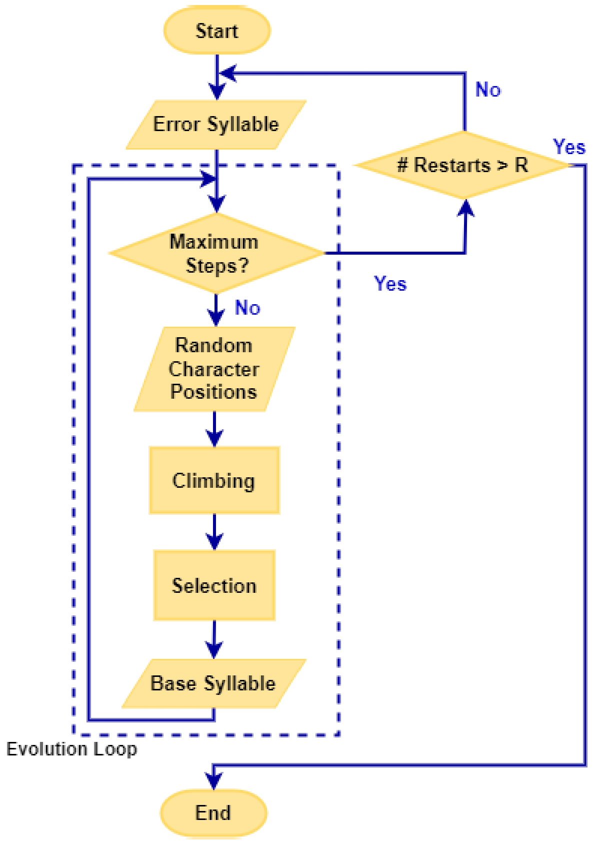


Figure 5. The candidate search process of the HC-based model.

1) In Step 1, the error patterns at one or more starting positions of s_e (maximum d positions) are substituted with the correction patterns obtained from the D_{cor} dictionary to generate a neighbor candidate s_{c_1}. Each correction pattern for an error pattern is randomly selected from the D_{cor} dictionary. If the neighbor candidate s_{c_1} occurs in any of the n-gram dictionaries, it becomes the correction candidate; otherwise, it is discarded.
2) In the following steps, we repeatedly substitute the error patterns in Step 1 with the other random correction patterns to generate the new neighbor candidates s_{c_i}. These neighbors are then checked to see if any of them become new correction candidates. Substitution operations for each error pattern are performed until the maximum number of correction patterns is reached. The maximum number of correction patterns is set to 300 in our experiments, as shown in Table 6. Additionally, we evaluate the performance of our proposed model with different values of the maximum number of correction patterns in Section IV-B3.

- Next, in the selection process, we consider two cases. In the first case, there exist the generated correction candidates among which the candidate achieving the highest objective score, $argmax_i(F_{obj}(s_{c_i}))$, defined in Section III-B3.b, is selected, and the algorithm continues to hill-climb from the highest-scored candidate. In the second case of no correction candidates found, the algorithm still continues from s_e. For both cases, the syllable from which we continue to hill-climb is called the **base syllable**. When the algorithm finds a base syllable, we consider the algorithm finishes an **evolution loop**. In the steps of the next evolution loop, we search for neighbor candidates by applying random correction pattern edits to the base syllable.
- At the start of the next evolution loop, d new character positions of the base syllable are randomly selected. Random correction patterns are then obtained to substitute the error patterns at the new positions of the base syllable. The new correction candidates and the new base syllable are produced by repeating the steps above. Similarly, the new base syllable will go under random correction pattern edits in the next loop.
- When the maximum number of steps is reached, the algorithm restarts with newly random character positions of s_e. The last base syllable found in each restart is recorded. The algorithm will continue until the predefined number of restarts is completed. The highest-scored base syllable among the last base syllables of all the restarts is selected as the final correction candidate of the error syllable, s_e.

In the search process above, the algorithm possibly gets evolved from a wrong base syllable due to the wrong character position selected at the start of the algorithm (not the position at which an actual error pattern is located). We prevent this in two ways: selecting random starting positions along the characters of the base syllable at the beginning of evolution loops and at the restart of the algorithm after it completes a certain number of steps.

Following our study [47] on the VNOnDB-OCR dataset, the errors with edit distances 1 and 2 contribute more than 80% of the total errors whereas the ones with the edit distance 1 account for over 60%. In addition, it is shown in [48] that the edit distances of 1 and 2 obtain better correction results instead of higher edit distances. Therefore, we apply the maximum number of edit operations as 2 (equivalently, $d = 2$) used in each step by selecting two random character positions of the base syllable at the start of each evolution loop. In other words, one or two correction pattern edits can take place at either or both of the two positions of the base syllable in each step of the evolution loop.

b: CANDIDATE RANKING AND SELECTION

Correction candidates are scored using an objective function. The objective function is a weighted sum of linguistic features extracted from each correction candidate and its context syllables. We adopt and modify several important word-level features which have been successfully employed for the recent OCR post-processing approaches, including word similarity, and word n-gram context frequency [15], [17], [18], [19], [21], [28], [49]. In addition, we propose a new linguistic feature at the character level, called pattern edit frequency. These features capture the diverse characteristics of the language. They are described as follows.

- **Bigram context frequency:**
Given an error syllable s_e, the bigram context frequency of the candidate s_{c_i}, denoted as $B(s_{c_i})$, is computed as the frequency sum of its left and right bigram contexts $s_{c_{i-1}}s_{c_i}$ and $s_{c_i}s_{c_{i+1}}$, and normalized regarding the maximum among the bigram context frequencies of all the candidates.
$$B(s_{c_i}) = \frac{f_2(s_{c_{i-1}}s_{c_i}) + f_2(s_{c_i}s_{c_{i+1}})}{max_{s'_{c_i}\in\mathbb{W}}(f_2(s_{c_{i-1}}s'_{c_i}) + f_2(s'_{c_i}s_{c_{i+1}}))}, \quad (1)$$
where $\mathbb{W}$ is the candidate set of s_e, and $f_2()$ is the bigram frequency-count function for the bigram dictionary.
- **Trigram context frequency:**
Similarly, the trigram context frequency of the candidate s_{c_i}, denoted as $T(s_{c_i})$, is computed as the frequency sum of its trigram contexts $s_{c_{i-2}}s_{c_{i-1}}s_{c_i}$, $s_{c_{i-1}}s_{c_i}s_{c_{i+1}}$, and $s_{c_i}s_{c_{i+1}}s_{c_{i+2}}$, and normalized regarding the maximum among the trigram context frequencies of all the candidates.
$$T(s_{c_i}) = \frac{\sum_{n=1}^{3} f_3(trigram_n(s_{c_i}))}{max_{s'_{c_i}\in\mathbb{W}}(\sum_{n=1}^{3} f_3(trigram_n(s'_{c_i})))}, \quad (2)$$
where $\mathbb{W}$ is the candidate set of s_e, $f_3()$ is the trigram frequency-count function for the trigram dictionary, and $trigram_n(s_{c_i})$ is the n-th trigram of the candidate s_{c_i}.

- **Syllable similarity:**
  The syllable similarity between the correction candidate s_c and the error syllable s_e, denoted as $S(s_c, s_e)$, measures the similarity between them. It is calculated as a weighted sum of the Normalized Longest Common Subsequence (NLCS) and the variations of Normalized Maximal Consecutive Longest Common Subsequence (NMCLCS) proposed in [15].

$$\begin{aligned} S(s_c, s_e) = {} & \alpha_1 \cdot NLCS(s_c, s_e) \\ & + \alpha_2 \cdot NMCLCS_1(s_c, s_e) \\ & + \alpha_3 \cdot NMCLCS_n(s_c, s_e) \\ & + \alpha_4 \cdot NMCLCS_z(s_c, s_e), \end{aligned} \tag{3}$$

  where NMCLCS$_1$, NMCLCS$_n$, and NMCLCS$_z$ are variations of NMCLCS starting from the first character, from any character, and ending at the last character, respectively. Additionally, $\alpha_1, \alpha_2, \alpha_3$, and α_4 are weights and have the sum of 1. They are set equal to 0.25 as in the related papers [15], [50] to keep the system unsupervised. It is because the corresponding variations of the Longest Common Subsequence have the same role in measuring the similarity of two strings.
- **Pattern edit frequency:**
  The pattern edit frequency is based on Algorithm 2 of correction pattern extraction. The pattern edit frequency of the candidate s_c, denoted as $P_{cor}(s_c)$, is determined as the frequency product of the correction patterns substituting for error patterns to transform s_e into s_c, and normalized with regard to the maximum among the pattern edit frequencies of all the candidates.

$$P_{cor}(s_c) = \frac{\prod_{p \in \mathbb{E}_{s_c}} f_{cor}(p)}{max_{s'_c \in \mathbb{W}}(\prod_{p' \in \mathbb{E}_{s'_c}} f_{cor}(p'))}, \tag{4}$$

  where each p is a correction pattern in the correction pattern set $\mathbb{E}_{s_c}$ used to transform s_e into s_c, each p' is a correction pattern in the correction pattern set $\mathbb{E}_{s'_c}$ used to transform s_e into s'_c, and $f_{cor}()$ is the correction pattern frequency-count function with respect to the correction pattern dictionary.

The objective function is used to compute the objective score of the correction candidate, s_c, and given as

$$\begin{aligned} F_{obj}(s_c) = {} & p_1 \cdot S(s_c, s_e) + p_2 \cdot B(s_c) \\ & + p_3 \cdot T(s_c) + p_4 \cdot P_{cor}(s_c), \end{aligned} \tag{5}$$

where $S(s_c, s_e)$, $B(s_c)$, $T(s_c)$, and $P(s_c)$ are the feature scores of syllable similarity, bigram context frequency, trigram context frequency, and pattern edit frequency of the correction candidate s_c, respectively. The weights p_1, p_2, p_3, and p_4 are subject to $p_1 + p_2 + p_3 + p_4 = 1$.

Since the weights sum up to 1, the objective score of the correction candidate s_c is in the range [0, 1] and considered as the confidence or probability of substituting s_e with s_c. The closer the objective score of a correction candidate is to 1, the more confident this correction candidate is used to substitute for the error syllable. A correction candidate with the highest objective score is selected as the final correction for s_e.

In our experiments, we heuristically examine different combinations of these weights to evaluate the performance of the proposed algorithm and to find optimal combinations.

To illustrate how to calculate the objective score of a correction candidate using the objective function based on linguistic features, we use a specific example as follows. Given the OCR error syllable "đảo", it is assumed that the candidate search process finds four correction candidates, and the corresponding character pattern edits of each correction candidate are "để" ("o"→"n", "ẩn"→"ể"), "đẩy" ("o"→"y"), "được" ("ả"→"ư", "o"→"ợc"), and "báo" ("ả"→"á", "đ"→"b"). For example, two pattern edits "o"→"n" and "ẩn"→"ể" are applied in such order to correct "đảo" to "để"; particularly, the edit "o"→"n" is selected to transform "đảo" into "đẩn", then "ẩn" is substituted by "ể" to transform "đẩn" into "để". The set of correction candidates for the error syllable "đảo" is $\mathbb{W}$ = {"để", "đẩy", "được", "báo"}. In fact, the number of correction candidates discovered in each evolution loop (or the number of base syllables found through the restarts) for each OCR error syllable is much larger in our experiments; here, we only use four correction candidates to simplify the calculation and illustration. The context syllables right preceding and following the error syllable "đảo" are "ngoại", "vừa", "dành", and "tiền" (the original GT text is "bà ngoại vừa <u>để</u> dành tiền mua được nền nhà" that is translated into English as "Grandma just saved money to buy a house" and the corresponding OCR text is "bà ngoại vừa <u>đảo</u> dành tiền mua được nền nhà"). In case the correction candidate "để" is selected, the bigram contexts of "để" are "vừa để" and "để dành", while the trigram contexts are "ngoại vừa để", "vừa để dành", and "để dành tiền". Table 5 shows the feature scores and objective scores of these correction candidates using Equations 1, 2, 3, 4, and 5. Based on the objective scores obtained, the candidate "để" is chosen as the final correction candidate for the error word "đảo" because it has the highest objective score.

IV. EXPERIMENTAL SETTINGS AND RESULTS

A. EXPERIMENTAL SETTINGS AND EVALUATION METRICS

The proposed HC-based model has the experimental settings for the model parameters such as the number of restarts, the number of steps, the maximum number of random correction patterns, the edit distance, and the objective function as shown in Table 6.

In our implementation, when a final correction suggestion (highest-scored base syllable) for an error syllable in the OCR text is found, it will replace the error syllable, and is then used as the context syllable to correct the next error syllables in the same OCR text. This helps further improve the OCR error correction performance.

The proposed models for the Vietnamese handwritten text recognition and correction competition are evaluated using two assessment metrics, the character error rate (CER) and

Table 5. **Feature scores and objective scores of correction candidates.**

Correction candidate	Syllable similarity	Bigram context frequency	Trigram context frequency	Pattern edit frequency	Objective score
để	0.3	0.353	1.0	$9.76 \cdot 10^{-5}$	0.421*
đẩy	0.5	0.017	0.2	1.0	0.393
được	0.214	1.0	0.2	$9.76 \cdot 10^{-5}$	0.347
báo	0.25	0.101	0.2	$9.76 \cdot 10^{-5}$	0.185

* Using the weight combination (p_1, p_2, p_3, p_4) as (0.5, 0.2, 0.2, 0.1), which is experimented and selected in Section IV-B1.

Table 6. **Parameter settings of the HC-based model.**

Parameter	Value	Description
Number of restarts (R)	*60*	The number of restarts the algorithm takes when the maximum number of steps is reached. At each restart, the algorithm starts again from random character positions of the error syllable.
Number of steps (S)	*100000*	The maximum number of steps the algorithm takes to search correction candidates for an error syllable.
Number of correction patterns	*300*	The maximum number of random correction patterns selected to substitute for each error pattern in an evolution loop.
Edit distance (d)	*2*	The maximum number of edit operations applied to a base syllable to generate a new candidate in each step.
Objective function	$F_{obj}(s_c)$	The scoring function of the adopted linguistic features.

word error rate (WER), which are commonly employed to measure transcription results in recognition systems [32], [51], [52].

Let $\mathbb{G}$ be the set of source texts (ground truths) in the VNOnDB-OCR dataset and let $\mathbb{C}$ be the set of corresponding corrected texts resulting from the OCR error correction. The normalized edit distance (NED) is given as

$$NED(G_i, C_i) = \frac{100}{|G_i|} \cdot d_{LV}(G_i, C_i), \tag{6}$$

where G_i is the i-th source text in $\mathbb{G}$, C_i is the corresponding corrected text of G_i, and d_{LV} is the Levenshtein (LV) edit distance [53] computed between the two texts G_i and C_i. The edit distance is normalized by the length of the source text G_i to avoid any bias related to the lengths of source texts.

The average normalized edit distance (ANED) for the entire dataset is then computed by

$$ANED(\mathbb{G}, \mathbb{C}) = \frac{1}{N} \sum_{i=1}^{N} NED(G_i, C_i), \tag{7}$$

where N is the number of source texts in $\mathbb{G}$.

For CER, the LV edit distance is computed on the character level (the minimum number of character edits between two texts). For WER, the LV edit distance is computed on the word level (the number of dissimilar words between two texts). CER and WER are the inverse performance metrics. In other words, a low value shows high performance and vice versa.

B. EXPERIMENTAL RESULTS AND DISCUSSIONS

Most of the approaches for the VOHTR competition [32] and the later ones [5], [28], [39] for the text recognition and correction of the VNOnDB database [4] include the post-processing step following the text recognizers.

There were the Google research team, IVTOV team, and MyScript team participating in the VOHTR competition. The Google team applies the basic preprocessing to the ink with scaling, normalizing, resampling, and representing by Bezier curves. The preprocessed data is then passed through multiple BiLSTM layers. The resulting output of the BiLSTM layers is post-processed with the *n*-gram language models at character and word levels using their private corpus. For the next participant, the IVTOV team employs line segmentation before preprocessing. The extracted online features are then used to train a recognition network of two BiLSTM layers with 100 cells in each layer. In the post-processing step, they apply the dictionary constraints to the output sequence of the recognition neural network. Another online recognition system is MyScript. The MyScript system preprocesses the digital ink by normalizing with a Bezier approximation and correcting the slope and slant. It comprises two recognizers, one is the feed-forward network for predicting characters from segmented candidates, and one is the BiLSTM network for predicting output text without segmentation. For post-processing, it uses a syllable-based *n*-gram language model trained on a large corpus (35 million tokens) including VTB, additional corpora, and lexica.

Regarding our recent approaches for OCR text correction on the same dataset, our statistical language model (SLM) [28] uses the *n*-gram language and error models which are based on the statistics of the linguistic features and OCR error characteristics extracted from the external corpus and the training data. Besides, our syllable-level NMT model [39] employs the BiLSTM-based encoder and decoder with the attention mechanism to suggest correction candidates for error syllables.

The text recognition and correction results of the participating approaches are shown in Table 7. Our proposed HC-based model achieves the CER of 4.13% and the WER of 9.47%. It can be seen that our proposed model outperforms most of the other approaches, e.g., the Google (GoogleTask2), the IVTOV (IVTOVTask2), the two-layered BiLSTM model [4], the AED with the DenseNet encoder [5], as well as our previous SLM and NMT models. Our proposed model helps

Table 7. **Text recognition and correction performance on the VNOnDB-OCR dataset [32].**

Method	Language model	Corpus	CER (%)	WER (%)
GoogleTask2	Character-based and word-based n-gram	Other	6.86	19.00
IVTOVTask2	Dictionary	VTB	3.24	14.11
MyScriptTask2_1	Syllable-based trigram	VTB	1.02	2.02
MyScriptTask2_2	Syllable-based trigram	VTB+ Others	1.57	4.02
Two-layered BiLSTM model on online features [4]	None	None	7.17	NA
AED model with DenseNet encoder [5]	None	None	4.67	13.33
Our previous SLM model [28]	Syllable-based n-gram and character edits	VTB	4.17	9.82
Our previous NMT model [39]	Syllable-based unigram and BiLSTM	VTB	NA	11.5
Proposed HC-based model	**Syllable-based n-gram and correction patterns**	**VTB**	**4.13** [b]	**9.47** [b]

[b]Using the weight combination (p_1, p_2, p_3, p_4) as (0.5, 0.2, 0.2, 0.1).

improve the CER and WER rates of the baseline AED model with the DenseNet encoder by about 0.5% and 3.9% respectively. The MyScript system (MyScriptTask2_1) has the best performance with the lowest CER at 1.02% and WER at 2.02%. However, our model only focuses on post-processing the OCR texts with high error rates (CER 4.67% and WER 13.33%), which are generated by the OCR process using the AED model with the DenseNet encoder. On the other hand, the MyScript approach, including both the text recognition and correction steps, first recognizes the online handwritten texts and then post-processes the recognized texts. The MyScript approach utilizing the online features from online handwritten patterns would produce better recognition results than the AED model with the DenseNet encoder, which is based on the converted images of online handwritten texts and can not exploit the online handwriting features. Moreover, Myscript used a syllable-based trigram language model that has been trained on a larger corpus (35 million tokens), including the VTB corpus and other corpora (compared to our model trained on the VTB corpus only).

Tables 8 and 9 show correct and incorrect candidate suggestions for the OCR errors in the test dataset by our HC-based model. The candidate suggestion for each OCR error syllable is the highest-scored base syllable among the last base syllables resulting from the restarts as described in Section III-B3.a. It is also illustrated by Table 5 how the correction candidates are scored and selected as final candidate suggestions in our proposed model.

In Table 8, the different types of OCR errors are detected and suggested with correct candidates, e.g., non-syllable errors (ID's 1-6), case-sensitive errors (ID 7), real-syllable errors (ID's 8-12), and multiple errors in the same text line (ID's 13-15).

On the other hand, incorrect candidates are suggested in Table 9 for non-syllable errors (ID's 1-3), real-syllable errors

Table 8. **Examples of correct candidate suggestions by the HC-based model.**

ID	OCR error	Ground truth	Candidate suggestion	OCR text line
1	nghẹ	nghẹn	nghẹn	Tuấn|nói|**nghẹ**|ngào
2	đẩo	để	để	nuôi|bà|ngoại|vừa|**đẩo**|dành|tiền|mua|được|nền|nhà.
3	đệnh	định	định	nhưng|tôi|khẳng|**đệnh**|là|mình|không|có
4	thuệ	Huệ	Huệ	ĐDV|**thuệ**|đem|chiếc|băng|ca|đẩy|vào|nhà|có|nạn|nhân.
5	dức	chức	chức	Vụ|việc|đang|được|cơ|quan|**dức**|năng|tiếp|tục|điều|tra|làm|rõ
6	karokee	karaoke	karaoke	hát|**karokee**|cũng|là|chuyện|bình|thường,
7	khoán	Khoán	Khoán	Hải|là|người|đã|thuê|Hoàng|Văn|**khoán**
8	sa	su	su	rừng|cao|**sa**|ở|Dầu|Giây
9	trọng	trạng	trạng	nhưng|cũng|chính|bộ|phận|này|của|VKS|ra|cáo|**trọng**
10	lát	Cát	Cát	Săn|chim|ở|**lát**|Bà
11	Bên	km	km	"|điểm|đen|"|ở|**Bên**|95
12	bia	bữa	bữa	ngoài|chợ|để|lo|**bia**|cơm|cho|thợ.
13	cới tràng	cởi tráng	cởi tráng	**cới**|trần|cùng|thanh|niên|trai|**tràng**
14	Cbó sẵn	Có sẵn	Có sẵn	**Cbó**|người|cho|tiền|mua|**sẫn**|bia|đá|để|đó,
15	Hăn Hỹ	Văn Vỹ	Văn Vỹ	Ông|Hồ|**Hăn**|Tiểu|,|xã|Ô|Long|**Hỹ**,

(ID's 4-6), and consecutive error syllables (ID's 7, 8). One reason for incorrect candidate suggestions is that due to the random selection of correction patterns in the D_{cor} dictionary, expected correction patterns for error patterns of an OCR error are not possibly selected; therefore, the correct candidate for the OCR error is not generated during the search process. We can overcome it by increasing the maximum number of correction patterns or increasing the number of restarts to improve the possibility of expected correction patterns to be selected. However, this will trade-off for the computational time. Another problem we also observe is that the correct candidate for an OCR error is indeed generated during the candidate search process, but it has a lower objective score than the base syllable, hence it is discarded and not selected as the final candidate. For example, the correct candidate (ground truth) "thì" is generated for the real-syllable error "thi" (ID 5) in an evolution loop, but the other generated candidate "di" achieves the highest objective score and is selected as the final candidate suggestion for "thi". For the consecutive error syllables (ID's 7, 8), the context syllables of the error syllable are also erroneous; as a result, the n-gram context frequency features do not make any contribution to the objective score.

From the experimental results, it can be derived that the correction pattern dictionary has better coverage of correction patterns than the character edit table (CET) used in our previous SLM model [28], where correction patterns are learned from the aligned GT-OCR texts of the same training dataset. Particularly, in the HC-based model, each error pattern of an OCR error corresponds to any possible correction pattern

Table 9. Examples of incorrect candidate suggestions by the HC-based model.

ID	OCR error	Ground truth	Candidate suggestion	OCR text line
1	chủy	khủng	hủy	nỗi|**chủy**|khiếp
2	Câân	Bên	Cân	|**Câân**|trong|có|bảng|liệt|kê| danh|sách|những|người|đóng| góp,
3	UBXS	VKS	UBND	hiện|đang|công|tác|tại|**UBXS**
4	tòm	tòa	mà	**tòm**|còn|có|lý|do|các|lời|khai,
5	thi	thì	đi	Bần|cùng|lắm|**thi**|làm|thêm| kiếm|đồng|tiền,
6	thẩm	phẩm	thi	|bà|Tám|phải|mua|chịu|thực| **thẩm**
7	khúng khiến	khủng khiếp	khúng khiến	Thật|là|**khúng**|**khiến**!
8	quản bao	quán bar	quản vào	các|**quản**|**bao**,

Table 10. Error correction results of the HC-based model for the different weight combinations.

ID	Weight combination (p_1, p_2, p_3, p_4)	CER (%)	WER (%)
1	(0.2, 0.2, 0.3, 0.3)	4.7427 [c]	10.8937
2	(0.2, 0.2, 0.4, 0.2)	4.4545	9.8186
3	(0.2, 0.2, 0.5, 0.1)	4.6376	10.0623
4	(0.2, 0.3, 0.2, 0.3)	5.2086	12.0345
5	(0.2, 0.3, 0.3, 0.2)	4.4177	9.8496
6	(0.2, 0.3, 0.4, 0.1)	4.6700	10.1203
7	(0.2, 0.4, 0.2, 0.2)	5.0003	11.1915
8	(0.2, 0.4, 0.3, 0.1)	4.7678	10.2015
9	(0.2, 0.5, 0.2, 0.1)	4.7911	10.2517
10	(0.3, 0.2, 0.2, 0.3)	5.1413	11.9842
11	(0.3, 0.2, 0.3, 0.2)	4.3252	9.7722
12	(0.3, 0.2, 0.4, 0.1)	4.5856	9.9501
13	(0.3, 0.3, 0.2, 0.2)	4.7579	10.9208
14	(0.3, 0.3, 0.3, 0.1)	4.5712	9.9192
15	(0.3, 0.4, 0.2, 0.1)	4.5820	9.9849
16	(0.4, 0.2, 0.2, 0.2)	4.6466	10.8047
17	(0.4, 0.2, 0.3, 0.1)	4.3104	9.5521
18	(0.4, 0.3, 0.2, 0.1)	4.2848	9.6601
19	(0.5, 0.2, 0.2, 0.1)	**4.1256**	**9.4714**

[c]The numbers are rounded to four decimal places.

in the D_{cor} dictionary, instead of only using a correction pattern set matched with each error pattern in the *CET* table. Therefore, the HC-based model is able to correct more OCR errors and obtains better error correction performance than the SLM model.

Moreover, as presented in Section III-B3.a, the HC algorithm is improved in two ways: selecting newly random character positions of the base syllable when starting an evolution loop and when restarting the algorithm after it finishes. These improvements help the algorithm get evolved from the base syllable at correct character positions (where actual error patterns are located) to generate expected candidates and, as a result, increase the correction performance. In other words, the improved HC algorithm avoids producing unexpected candidates (local optimal solutions) from wrong character positions (not the positions at which actual error patterns are located). Through efficient and appropriate parameter settings, our proposed model can provide high-quality candidate generation and error correction.

In addition to the Vietnamese language illustrated and experimented with in this study, which belongs to the Latin script language family and involves diacritic marks, our proposed algorithm and model can be used for other Latin script languages (e.g., English, French…) since the proposed correction pattern extraction algorithm, the error detection model based on the *n*-gram dictionaries, and the error correction model using the adapted HC-based candidate search process and correction patterns are also applicable to these languages. Furthermore, because our proposed model can be applied directly to the erroneous OCR texts and does not depend on any specified parameters of the OCR system, it is applicable not only to the OCR texts of Vietnamese handwriting but also to any Vietnamese OCR texts regardless of OCR systems. In general, it can also be employed for any texts that contain spelling and linguistic errors in Vietnamese or in other Latin script languages.

Last but not least, the proposed model has two advantages. Firstly, there is no need to manually align GT-OCR texts of the training dataset in the preprocessing step as in the supervised approaches. Secondly, it has better coverage of correction patterns as discussed above. On the contrary, as shown in our previous studies [21], [28], the *CET* table constructed from the training dataset of aligned GT-OCR texts might not cover new character patterns that only exist in the test dataset. Thus, OCR errors in the test dataset containing these new error patterns might not be corrected properly.

1) WEIGHT COMBINATIONS

In reference to the feature weights specified in Section III-B3.b, it is necessary to set up more experiments to find best performing weight combinations of (p_1, p_2, p_3, p_4), with which the OCR error correction obtains the lowest CER and WER error rates. Because of limited computational resources, we only consider an incremental change of each weight for 10%; for example, each weight will take a value in the set of $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. This also does not lose generality since similar results can be obtained with finer weight increments.

The proposed model does not make use of OCR error characteristics; in other words, it relies more on the linguistic features of syllable similarity, bigram and trigram contexts. Therefore, we will set the weights of syllable similarity, bigram context frequency, and trigram context frequency to higher values. As a result, the following settings are considered: $p_1 = \{0.2, 0.3, 0.4, 0.5\}$, $p_2 = \{0.2, 0.3, 0.4, 0.5\}$, $p_3 = \{0.2, 0.3, 0.4, 0.5\}$, and $p_4 = \{0.1, 0.2, 0.3\}$. Since each possible combination of weights leads to a different configuration for the HC algorithm, there is a total of 19 weight combinations under the weight constraint $p_1 + p_2 + p_3 + p_4 = 1$. The goal is to find weight combinations that optimize the performance of the HC algorithm. The error correction results for these 19 weight combinations are shown in Table 10.

In Table 10, the combinations with the weight of syllable similarity feature (p_1) equal to 0.4 or 0.5 (four combinations of the ID's 16-19 except the ID 16) obtain better correction

Table 11. Error rate statistics of the HC-based model for the (0.5, 0.2, 0.2, 0.1) weight combination.

ID	CER (%)	Mean	Variance	WER (%)	Mean	Variance
1	4.0873	4.1256	0.00099	9.3932	9.4714	0.00205
2	4.1412			9.4667		
3	4.1143			9.4783		
4	4.1017			9.4397		
5	4.1600			9.5286		
6	4.1322			9.4745		
7	4.1286			9.4938		
8	4.0793			9.4165		
9	4.1295			9.4861		
10	4.1816			9.5363		

results than the remaining combinations (ID's 1-15) according to the CER and WER metrics. Furthermore, in each combination group with the same weight p_1 (e.g., ID's 1-9, ID's 10-15, and ID's 16-18), the combinations with a lower weight value of the pattern edit frequency feature ($p_4 = 0.1$) show more effective since they tend to have lower error rates compared to the other ones with the higher value of p_4 as 0.2 or 0.3. Besides, the weight combinations of ID's 17-19, where p_1 is set to 0.4 or 0.5 and p_4 is 0.1, get better WER than that of our previous SLM model (9.82%) and competitive CER to the SLM model (4.17%).

It is also observed that the n-gram context frequency features (p_2 and p_3 weights) have a similar contribution to the objective score. The weight combination (0.5, 0.2, 0.2, 0.1) achieving the highest correction performance (lowest CER and WER rates) will be selected in the next experiments for verification and validation of randomness and search-based parameters of the proposed model.

2) RANDOMNESS

Regarding the randomness added to the HC algorithm, we conduct the same experiment with the weight combination (0.5, 0.2, 0.2, 0.1) ten times and verify the mean and variance of the CER and WER error rates (see Table 11). We can see that the CER and WER rates are close to the mean values (rounded as 4.13% and 9.47% respectively, which are used in Table 7), and the variances are correspondingly small. This reflects that our HC-based model suggests almost the same correction candidates for the error syllables in the different experiments.

Moreover, for further check of the randomness and stability of the proposed HC algorithm, we conduct one additional group of ten experiments for the weight combination (0.5, 0.2, 0.2, 0.1) and compare its CER and WER rates with those of the experiment group in Table 11 using Wilcoxon signed rank test [54]. The error rates for this additional experiment group are given in Table 12. It is indicated that the results are statistically similar between the experimental groups. In particular, for the two experimental groups of the same configuration (0.5, 0.2, 0.2, 0.1) in Tables 11 and 12, the p-value regarding the CER rates is computed as 0.7695 which is greater than the significance level of 0.05; similarly, it is 0.7596 between the WER rates of the two groups. According

Table 12. Error rate statistics of the additional experiment group.

ID	Weight combination (0.5, 0.2, 0.2, 0.1)	
	CER (%)	WER (%)
1	4.1116	9.4242
2	4.1394	9.5015
3	4.1268	9.4629
4	4.1457	9.5325
5	4.1125	9.4474
6	4.0945	9.4242
7	4.1672	9.5247
8	4.1322	9.4745
9	4.1493	9.5209
10	4.1008	9.4551

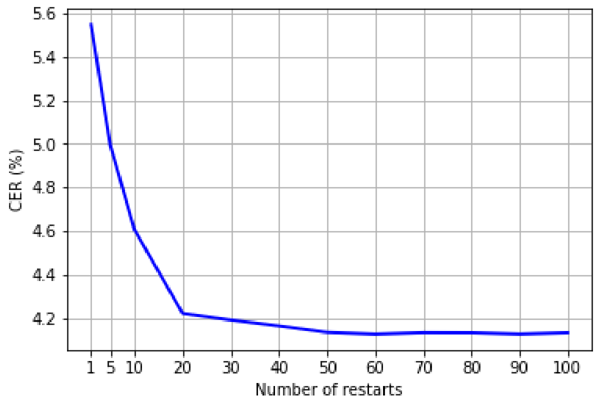
to the above statistics, our proposed HC algorithm is shown to be stable with high confidence under the parameter settings in Table 6.

3) SEARCH-BASED PARAMETERS

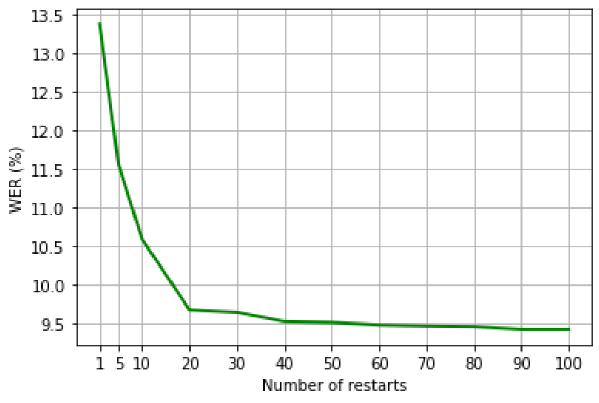
In order to evaluate the effects of two search-based parameters on error correction performance, including the number of restarts and the maximum number of correction patterns, we repeat the experiments with different values of the number of restarts and the maximum number of correction patterns. For example, we examine the values of the number of restarts within the set {1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100} while keeping the other parameters the same as in Table 6. Similarly, we change the maximum number of correction patterns with the values in the set {50, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000} (the size of the correction pattern dictionary D_{cor} is 1928). The error correction results of the proposed model for these two search-based parameters are depicted in Figures 6 and 7. The weight combination (0.5, 0.2, 0.2, 0.1) is used for these experiments.

In Figure 6, we can observe that the CER and WER curves decline dramatically, meaning the error correction performance gains much improvement, when the number of restarts increases to 20. In addition, with the number of restarts higher than 60 (e.g., 70, 80, 90, 100), the error correction performance almost does not improve concerning the CER and WER rates in Figures 6a and 6b respectively. In other words, when the number of restarts reaches 60, the search process is able to select the right error character positions of each OCR error and apply most of the correction patterns in the D_{cor} dictionary to substitute error patterns and generate expected correction candidates. Furthermore, increasing the number of restarts will cost more computational time. Thus, the number of restarts as 60 is an appropriate choice for the OCR error correction task on the VNOnDB-OCR dataset.

In Figure 7, it is interesting to see that for the low number of correction patterns (e.g., 50, 100, 200), the error correction results are quite close to those for the number of correction patterns equal to 300 on both the CER and WER rates (comparative to 4.13% and 9.47% respectively). It is possible that with the settings of the other parameters as in Table 6, e.g., the number of steps to 100000 and the number of restarts to 60, the HC algorithm can perform a large enough amount of

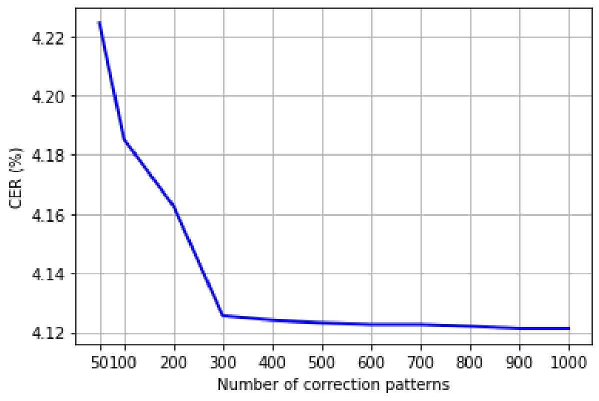


(a) CER

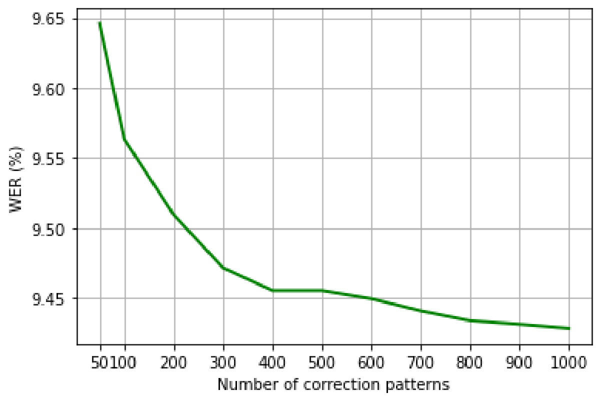


(b) WER

Figure 6. The error rates versus the numbers of restarts.



(a) CER



(b) WER

Figure 7. The error rates versus the numbers of correction patterns.

evolution loops as well as explore most of the possible correction candidates for each OCR error even with lower values of the maximum number of random correction patterns in each evolution loop. For the number of correction patterns higher than 300, the CER and WER error rates get improved, but they decrease slowly. In addition, like the number of restarts, the computational cost will grow quadratically when we increase the number of correction patterns. It is because, regarding the HC-based candidate search algorithm, the total candidates explored in each evolution loop are equivalent to p^d, where p represents the maximum number of correction patterns in an evolution loop, and d stands for the edit distance. Since d is set to 2, p^d becomes p^2. In the HC-based model, we choose the number of random correction patterns as 300 to achieve high-quality candidate generation and keep the computational time from not growing too much.

According to the above experimental results, it can be concluded that the maximum number of correction patterns and the number of restarts should be set to appropriate values to ensure both the quality of the error correction task and the efficient computational cost. Indeed, there is a trade-off between the higher settings of the search-based parameters for better model performance and the computational cost.

4) COMPLEXITY

Regarding the candidate search process of the HC-based model, in each evolution loop, the proposed HC algorithm explores the candidate search space of a base syllable by applying random correction pattern edits at random character positions of the base syllable until the maximum number of correction patterns is reached. The complexity of the proposed HC algorithm is computed as follows:

- In each step, a neighbor candidate is generated by applying random correction pattern edits to a base syllable at d random character positions.
- Let S be the maximum number of steps that the HC algorithm takes to search for correction candidates of an error syllable until it restarts, called a cycle. The total number of candidates explored in a cycle of the algorithm is also S.
- When the algorithm restarts the number of times (R), the accumulative candidates explored until all the restarts finish are equivalent to $S \cdot R$.

In the experimental settings, we have $S = 100000$, and $R = 60$. For simplicity and without losing generality, we consider the parameters S and R as a constant value n. The complexity of the whole search operation of the HC algorithm is $O(n^2)$. The HC-based model has quadratic time complexity. However, in our experiments, the value of S is set much higher than R (e.g., the value 100000 of S compared to 60 of R). Furthermore, the HC-based model only produces one highest-scored base syllable as the final correction candidate for each error syllable. For extension, we can collect the last base syllables at the end of the restarts as the top candidate suggestions for each error syllable, with which the HC-based model can be considered as a semi-automatic approach that involves users selecting the most appropriate correction candidate among the top candidate suggestions.

V. CONCLUSION

In this paper, we propose a novel unsupervised approach for OCR error correction using an adapted hill climbing algorithm without knowledge of OCR error characteristics. Correction candidates for OCR errors are explored and generated by correction pattern edits, which are constructed from only the original GT texts in the training data. The experimental results show that our proposed model outperforms various baseline models in the ICFHR 2018 competition. Our model is capable of generating high-quality correction candidates for OCR errors and is stable with high confidence through efficient settings of the model parameters. In future work, instead of searching candidates from all possibly random correction patterns from the correction pattern dictionary that would take considerable search time, we will investigate and suggest rules for selecting more appropriate correction patterns from the correction pattern dictionary (e.g., based on vowel or consonant type of error patterns, or similar character glyphs) to reduce the number of correction patterns used to substitute for every error pattern and therefore further improve candidate search time for OCR errors.

References

[1] M. R. Gupta, N. P. Jacobson, and E. K. Garcia, "OCR binarization and image pre-processing for searching historical documents," *Pattern Recognit.*, vol. 40, no. 2, pp. 389–397, Feb. 2007.

[2] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 5, pp. 855–868, May 2009.

[3] A. A. Desai, "Gujarati handwritten numeral optical character reorganization through neural network," *Pattern Recognit.*, vol. 43, no. 7, pp. 2582–2589, Jul. 2010.

[4] H. T. Nguyen, C. T. Nguyen, P. T. Bao, and M. Nakagawa, "A database of unconstrained Vietnamese online handwriting and recognition experiments by recurrent neural networks," *Pattern Recognit.*, vol. 78, pp. 291–306, Jun. 2018.

[5] A. D. Le, H. T. Nguyen, and M. Nakagawa, "An end-to-end recognition system for unconstrained Vietnamese handwriting," *Social Netw. Comput. Sci.*, vol. 1, no. 1, pp. 1–8, Jan. 2020.

[6] G. Chiron, A. Doucet, M. Coustaty, M. Visani, and J.-P. Moreux, "Impact of OCR errors on the use of digital libraries: Towards a better access to information," in *Proc. ACM/IEEE Joint Conf. Digital Libraries (JCDL)*, Sep. 2017, pp. 249–252.

[7] G. T. Bazzo, G. A. Lorentz, D. Suarez Vargas, and V. P. Moreira, "Assessing the impact of OCR errors in information retrieval," in *Proc. 42nd Eur. Conf. IR Res. (ECIR)*, Lisbon, Portugal: Springer-Verlag, Apr. 2020, pp. 102–109.

[8] E. Mittendorf and P. Schäuble, "Information retrieval can cope with many errors," *Inf. Retr.*, vol. 3, no. 3, pp. 189–216, 2000.

[9] H. Jing, D. Lopresti, and C. Shih, "Summarization of noisy documents: A pilot study," in *Proc. HLT-NAACL Text Summarization Workshop*, vol. 5. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 25–32.

[10] A. Hamdi, A. Jean-Caurant, N. Sidere, M. Coustaty, and A. Doucet, "An analysis of the performance of named entity recognition over OCRed documents," in *Proc. ACM/IEEE Joint Conf. Digit. Libraries (JCDL)*, Jun. 2019, pp. 333–334.

[11] D. Miller, S. Boisen, R. Schwartz, R. Stone, and R. Weischedel, "Named entity extraction from noisy input: Speech and OCR," in *Proc. 6th Conf. Appl. Natural Lang. Process.* Stroudsburg, PA, USA: Association for Computational Linguistics, 2000, pp. 316–324.

[12] X. Lin, "Impact of imperfect OCR on part-of-speech tagging," in *Proc. 7th Int. Conf. Document Anal. Recognit.*, 2003, pp. 284–288.

[13] M. Mieskes and S. Schmunk, "OCR quality and NLP preprocessing," in *Proc. Workshop Widening NLP*. Florence, Italy: Association for Computational Linguistics, 2019, pp. 102–105.

[14] G. Zu, M. Murata, W. Ohyama, T. Wakabayashi, and F. Kimura, "The impact of OCR accuracy on automatic text classification," in *Content Computing* (Lecture Notes in Computer Science), vol. 3309, C. H. Chi, K. Y. Lam, Eds. Berlin, Germany: Springer, 2004, pp. 403–409.

[15] A. Islam and D. Inkpen, "Real-word spelling correction using Google web 1T N-gram with backoff," in *Proc. Int. Conf. Natural Lang. Process. Knowl. Eng.* Hong Kong, China: Association for Computing Machinery, Sep. 2009, pp. 1689–1692.

[16] Y. Bassil and M. Alwani, "OCR post-processing error correction algorithm using Google's online spelling suggestion," *J. Emerg. Trends Comput. Inf. Sci.*, vol. 3, no. 1, 2012, pp. 90–99.

[17] I. Kissos and N. Dershowitz, "OCR error correction using character correction and feature-based word classification," in *Proc. 12th IAPR Workshop Document Anal. Syst. (DAS)*, Apr. 2016, pp. 198–203.

[18] J. Mei, A. Islam, A. Moh'd, Y. Wu, and E. Milios, "Statistical learning for OCR error correction," *Inf. Process. Manage.*, vol. 54, no. 6, pp. 874–887, Nov. 2018.

[19] 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 *Maturity and Innovation in Digital Libraries*, M. Dobreva, A. Hinze, and M. Zumer, Eds. Cham, Switzerland: Springer, 2018, pp. 278–289.

[20] G. Khirbat, "OCR post-processing text correction using simulated annealing (OPTeCA)," in *Proc. Australas. Lang. Technol. Assoc. Workshop*. Brisbane, QLD, Australia, 2017, pp. 119–123.

[21] Q.-D. Nguyen, D.-A. Le, N.-M. Phan, and I. Zelinka, "OCR error correction using correction patterns and self-organizing migrating algorithm," *Pattern Anal. Appl.*, vol. 24, pp. 701–721, Nov. 2021.

[22] H. Afli, Z. Qiu, A. Way, and P. Sheridan, "Using SMT for OCR error correction of historical texts," in *Proc. 10th Int. Conf. Lang. Resour. Eval. (LREC)*. Portoroz, Slovenia: European Language Resources Association (ELRA), 2016, pp. 962–966.

[23] H. Afli, L. Barrault, and H. Schwenk, "OCR error correction using statistical machine translation," *Int. J. Comput. Linguistics Appl.*, vol. 7, no. 1, pp. 175–191, 2016.

[24] S. Schulz and J. Kuhn, "Multi-modular domain-tailored OCR post-correction," in *Proc. Conf. Empirical Methods Natural Lang. Process.* Copenhagen, Denmark: Association for Computational Linguistics, 2017, pp. 2716–2726.

[25] C. Amrhein and S. Clematide, "Supervised OCR error detection and correction using statistical and neural machine translation methods," *J. Lang. Technol. Comput. Linguistics*, vol. 33, no. 1, pp. 49–76, Jul. 2018.

[26] T. T. H. Nguyen, A. Jatowt, N.-V. Nguyen, M. Coustaty, and A. Doucet, "Neural machine translation with BERT for post-OCR error detection and correction," in *Proc. ACM/IEEE Joint Conf. Digit. Libraries*. New York, NY, USA: Association for Computing Machinery, Aug. 2020, pp. 333–336.

[27] G. Chiron, A. Doucet, M. Coustaty, and J. Moreux, "ICDAR2017 competition on post-OCR text correction," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR)*, vol. 1, Nov. 2017, pp. 1423–1428.

[28] Q.-D. Nguyen, D.-A. Le, and I. Zelinka, "OCR error correction for unconstrained Vietnamese handwritten text," in *Proc. 10th Int. Symp. Inf. Commun. Technol. (SoICT)*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 132–138.

[29] D. Jurafsky and J. Martin, "Speech and language processing: An introduction to natural language processing," in *Computational Linguistics, and Speech Recognition*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 2008, ch. 5, pp. 42–47.

[30] T. Segaran and J. Hammerbacher, *Beautiful Data: The Stories Behind Elegant Data Solutions*. Sebastopol, CA, USA: O'Reilly Media, 2009, ch. 14, pp. 219–242.

[31] T.-D. Pham, Q.-D. Nguyen, D.-A. Le, N.-M. Phan, and P. Kromer, "Candidate word generation for OCR errors using optimization algorithm," in *Proc. 1st Int. Conf. Van Lang. Heritage Technol.*, vol. 2406, 2021, Art. no. 020028.

[32] H. T. Nguyen, C. T. Nguyen, and M. Nakagawa, "ICFHR 2018–competition on Vietnamese online handwritten text recognition using HANDS-VNOnDB (VOHTR2018)," in *Proc. 16th Int. Conf. Frontiers Handwriting Recognit. (ICFHR)*, Aug. 2018, pp. 494–499.

[33] M. Volk, L. Furrer, and R. Sennrich, "Strategies for reducing and correcting OCR errors," in *Proc. Lang. Technol. Cultural Heritage*. Berlin, Germany: Springer, 2011, pp. 3–22.

[34] J. Evershed and K. Fitch, "Correcting noisy OCR: Context beats confusion," in *Proc. 1st Int. Conf. Digit. Access Textual Cultural Heritage*, May 2014, pp. 45–51.

[35] I. Zelinka, "SOMA—Self-organizing migrating algorithm," in *New Optimization Techniques in Engineering*. Berlin, Germany: Springer, 2004, pp. 167–217.

[36] F. Fancellu, A. Way, and M. O'brien, "Standard language variety conversion for content localisation via SMT," in *Proc. 17th Annual Conf. Eur. Assoc. Mach. Transl.* Dubrovnik, Croatia: European Association for Machine Translation, 2014, pp. 143–149.

[37] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst, "Moses: Open source toolkit for statistical machine translation," in *Proc. 45th Annu. Meeting Assoc. Comput. Linguistics*. Prague, Czech Republic: Association for Computational Linguistics, 2007, pp. 177–180.

[38] G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. Rush, "OpenNMT: Open-source toolkit for neural machine translation," in *Proc. ACL Syst. Demonstrations*. Vancouver, BC, Canada: Association for Computational Linguistics, 2017, pp. 67–72.

[39] Q.-D. Nguyen, D.-A. Le, N.-M. Phan, P. Kromer, and I. Zelinka, "OCR error correction for Vietnamese handwritten text using neural machine translation," in *Proc. 1st Int. Conf. Van Lang. Heritage Technol.*, vol. 2406, 2021, Art. no. 020022.

[40] C.-D. Vu Hoang and A.-T. Aw, "An unsupervised and data-driven approach for spell checking in Vietnamese OCR-scanned texts," in *Proc. Workshop Innov. Hybrid Approaches to Process. Textual Data*. Avignon, France: Association for Computational Linguistics, 2012, pp. 36–44.

[41] S. C. Özcan and H. Kaya, "An analysis of travelling salesman problem utilizing Hill climbing algorithm for a smart city touristic search on OpenStreetMap (OSM)," in *Proc. 2nd Int. Symp. Multidisciplinary Stud. Innov. Technol. (ISMSIT)*, Oct. 2018, pp. 1–5.

[42] S. S. Skiena, *The Algorithm Design Manual*, 2nd ed. New York, NY, USA: Springer, 2008.

[43] S. A. Akramifar and G. Ghassem-Sani, "Fast forward planning by guided enforced Hill climbing," *Eng. Appl. Artif. Intell.*, vol. 23, no. 8, pp. 1327–1339, Dec. 2010.

[44] S. Chalup and F. Maire, "A study on Hill climbing algorithms for neural network training," in *Proc. Congr. Evol. Comput.-CEC*, 1999, pp. 2014–2021.

[45] A. Babadi, B. Omoomi, and G. Kendall, "EnHiC: An enforced Hill climbing based system for general game playing," in *Proc. IEEE Conf. Comput. Intell. Games (CIG)*, Aug. 2015, pp. 193–199.

[46] P.-T. Nguyen, X.-L. Vu, T.-M.-H. Nguyen, V.-H. Nguyen, and H.-P. Le, "Building a large syntactically-annotated corpus of Vietnamese," in *Proc. 3rd Linguistic Annotation Workshop ACL-IJCNLP*. Suntec, Singapore: Association for Computational Linguistics, 2009, pp. 182–185.

[47] Q.-D. Nguyen, D.-A. Le, N.-M. Phan, and I. Zelinka, "An in-depth analysis of OCR errors for unconstrained Vietnamese handwriting," in *Proc. 7th Int. Conf. Future Data Secur. Eng. (FDSE)*, in Lecture Notes in Computer Science, vol. 12466. Cham, Switzerland: Springer, 2020, pp. 448–461.

[48] Q.-D. Nguyen, N.-M. Phan, and P. Kromer, "OCR error correction for Vietnamese OCR text with different edit distances," in *Proc. 14th Int. Conf. Intell. Netw. Collaborative Syst. (INCoS)*, in Lecture Notes in Networks and Systems, vol. 527. Cham, Switzerland: Springer, 2022, pp. 130–139.

[49] Q.-D. Nguyen, D.-A. Le, N.-M. Phan, N.-T. Phan, and P. Kromer, "Statistical post-processing approaches for OCR texts," in *Proc. Int. Joint Conf. Adv. Comput. Intell.*, M. S. Uddin, P. K. Jamwal, J. C. Bansal, Eds. Singapore: Springer, 2022, pp. 457–467.

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

[51] P. Dreuw, G. Heigold, and H. Ney, "Confidence- and margin-based MMI/MPE discriminative training for off-line handwriting recognition," *Int. J. Document Anal. Recognit.*, vol. 14, no. 3, pp. 273–288, Sep. 2011.

[52] W. Swaileh, "Language modelling for handwriting recognition," in *Modeling and Simulation*. Normandy: Normandie Université, 2017.

[53] G. Navarro, "A guided tour to approximate string matching," *ACM Comput. Surveys*, vol. 33, no. 1, pp. 31–88, Mar. 2001.

[54] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 special session on real parameter optimization," *J. Heuristics*, vol. 15, no. 6, pp. 617–644, Dec. 2009.

QUOC-DUNG NGUYEN received the B.Eng. degree from the Posts and Telecommunications Institute of Technology, Vietnam, in 2003, the M.Eng. degree from the Catholic University of Leuven, Belgium, in 2010, and the Ph.D. degree from the VSB–Technical University of Ostrava, Czech Republic, in 2022. He is currently a Lecturer with Van Lang University, Vietnam. His research interests include natural language processing, machine learning, deep learning, and evolutionary algorithms.

NGUYET-MINH PHAN received the B.Sc. degree in information technology from the University of Science, Vietnam, in 2005, and the M.Sc. degree in computer science from the University of Information Technology, Vietnam, in 2010. She is currently a Lecturer with Sai Gon University, Vietnam. Her research interests include mobile applications, natural language processing, and data science.

PAVEL KRÖMER (Senior Member, IEEE) received the Ph.D. degree from the VSB–Technical University of Ostrava, Czech Republic, in 2010. He was a Postdoctoral Fellow with the University of Alberta, Edmonton, Canada, in 2014. From 2011 to 2016, he was a Researcher with the IT4Innovations National Supercomputing Center, VSB–Technical University of Ostrava, where he is currently an Associate Professor. His research interests include artificial intelligence, evolutionary computation, machine learning, and the real-world applications of intelligent methods.

DUC-ANH LE (Member, IEEE) received the B.Eng. degree (Hons.) in computer science from the Ho Chi Minh City University of Technology, and the master's and Ph.D. degrees from the Tokyo University of Agriculture and Technology, in 2014 and 2017, respectively. He has experience in leadership at a variety of academic and industrial projects on document analysis and recognition. He is currently a Project Assistant Professor with The Institute of Statistical Mathematics, Japan, and a Visiting Professor with Nguyen Tat Thanh University, Vietnam. His research interests include handwritten recognition, pattern recognition, and deep learning and their applications.