Adaptive Edit-Distance and Regression Approach for Post-OCR Text Correction

Thi-Tuyet-Hai Nguyen, Mickaël Coustaty, Antoine Doucet, Adam Jatowt, Nhu-Van Nguyen

Adaptive Edit-Distance and Regression Approach for Post-OCR Text Correction

Thi-Tuyet-Hai Nguyen[1(✉)], Mickael Coustaty[1], Antoine Doucet[1], Adam Jatowt[2], and Nhu-Van Nguyen[1]

[1] L3i, University of La Rochelle, La Rochelle, France
{hai.nguyen, mickael.coustaty, antoine.doucet, nhu-van.nguyen}@univ-lr.fr
[2] Department of Social Informatics, Kyoto University, Kyoto, Japan
adam@dl.kuis.kyoto-u.ac.jp

Abstract. Post-processing is a crucial step in improving the performance of OCR process. In this paper, we present a novel approach which explores a modified way of candidate generating and candidate scoring at character level as well as word level. These features are combined with some important features suggested by related work for ranking candidates in a regression model. The experimental results show that our approach has comparable results with the top performing approaches in the Post-OCR text correction competition ICDAR 2017.

Keywords: Post-OCR processing · Noisy channel · Language model Regression model

1 Introduction

Born-analog documents still contain massive knowledge which is valuable to our society. For the purpose of preservation and easier accessibility, a lot of efforts have been devoted to optical character recognition (OCR) systems to transform paper-based documents into digital form. However, poor physical quality of documents and limitations of text recognition techniques result in the low performance of OCR systems. Erroneous OCR-generated texts not only prevent users from retrieving relevant information but also cause reading difficulties. Post-processing is the last activity in OCR pipeline, attempting to detect and correct OCR errors.

Diverse approaches have been conducted for OCR post-processing. They can be divided into three main categories: manual error correction, dictionary-based error correction and context-based error correction [2].

The first type lets humans manually review and correct OCR-ed texts. It requires continuous manual intervention. Therefore, it is not only costly but also time-consuming and error-prone.

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

M. Dobreva et al. (Eds.): ICADL 2018, LNCS 11279, pp. 278–289, 2018.
https://doi.org/10.1007/978-3-030-04257-8_29

The second type uses a lookup dictionary to search for misspelled words and correct them automatically [17]. This kind of approach is easy to implement and use, however, it is unable to correct errors due to their grammatical and semantic contexts.

Finally, the context-based approach type is proposed to eliminate the disadvantages mentioned above. Most solutions of this type make use of the noisy channel model and statistical language model. Some approaches apply machine translation techniques which translate OCR output into corrected text in the same language. Some others combine different features in a prediction model to avoid bias and select a more accurate candidate from a pool of candidates.

The OCR post-processing consists of two parts: error detection and error correction. In this paper, we focus on the second part with the given list of known error positions. Our approach explores the noisy channel model, language model as well as other features for generating and ranking candidates using machine learning techniques. The evaluation results show that our method reaches comparable performance to the best performing teams in the English monograph dataset of the Post-OCR text correction competition ICDAR 2017 [4].

The remainder of this paper is organized as follows. In Sect. 2, we describe previous works related to OCR post-processing. Section 3 gives the detailed description of our approach. The experimental results are shown and discussed in Sect. 4. Finally, we give some conclusions in Sect. 5.

2 Related Work

The post-processing model detects and corrects misspellings of both non-words and real-words in the OCR-ed text. The literature of this research field contains a rich family of models, especially with the context-based type.

The dictionary-based type tries to correct misspelled words in isolation and does not take the context other nearby errors into consideration [17]. This type is simple to implement, but it cannot deal with real-word errors.

The context-based type considers grammatical and semantic contexts of errors and is more promising to correct such real-word errors. Most of the techniques of this type rely on noisy channel and language model. The others explore different machine learning techniques to suggest correct candidates.

Tong *et al.* [20] explored multiple features, including character n-grams, confusion probabilities, and word bi-gram language model to fix errors. Using some features similar to the ones in Tong *et al.*, WFST-PostOCR approach (the participant in the Post-OCR text correction competition) [4] applied the probabilistic character error model and language model. However, these models were compiled into weighted finite-state edit transducers (WFST). The best token sequence was the best path of WFST. This team achieved the third rank with 28% improvement, which confirmed the importance of error model and language model in correcting erroneous OCR-ed text.

Other promising approaches of competition, including Multi-Modular Domain-Tailored (MMDT), Character Level Attention Model (CLAM), Character-based Statistical and Neural Machine Translation (Char-SMT/NMT), are based on machine translation techniques.

MMDT [18] approach combined many modules from word level (Original words, Spell checker, Compounder, Word splitter, Text-Internal Vocabulary) to sentence level (Statistical Machine Translation) for candidate suggestion. Then, the decision module of Moses decoder [13] was used to rank candidates. In our opinion, Text-Internal Vocabulary function, which suggests high-frequency words in OCR-ed text as correction of errors with a small distance, easily leads to bias.

Two other competition approaches (Char-SMT/NMT, CLAM) rely on character based machine translation technique. There are however some limits of machine translation at character level. Character-level models enable to produce non-words which may be close to the reference. It is necessary to ensure that words generated by a character-level model should be valid words before ranking and suggesting them as relevant candidates. Tiedemann [19] suggested to include a string similarity measure as feature function. Furthermore, Afli *et al.* [1] demonstrated that machine translation at word level is better than at character level.

Other approaches applied a regression model in candidate ranking. Kissos *et al.* [12] extracted six features to train a linear regressor, including confusion probability with a single edition, unigram frequency, context feature (backward bi-gram frequency, forward bi-gram frequency), term frequency in the OCR-ed text, and word confidence. In our opinion, the unigram frequency feature is not a good feature because it does not take run-on errors into account, for example, for the error "doubtfud.of" the possible correct candidate is "doubtfull of" which is bi-gram instead of unigram. In addition, similar to internal vocabulary feature of MMDT, term frequency feature also easily causes bias. Furthermore, this approach did not consider an important feature used in real-word error correction [10], which is the similarity between error and candidate.

Mei *et al.* [15] also suggested to rank candidates using a predictive model. They extracted six features of each candidate. Besides word n-gram and candidate frequency features, they paid attention to string similarity which is missing from Kissos' approach. However, they ignored another important feature (confusion probability) which is used in several successful post-processing approaches [6, 12, 14, 20].

Our method belongs to the context-based approach type. We propose to make use of confusion probability obtained from the noisy channel model of multiple editions (instead of single edition), and context probability given by language model. Then, these two features and some essential features suggested by related works [12, 15] are used to train a regression model. The experimental results show that our multi-modular approach is comparable to the ones of the teams participating in the Post-OCR text correction competition ICDAR 2017.

3 Regression Approach for Post-OCR Text Correction

We explore all information related to error from characters constituting error to context words surrounding error to suggest correct candidates. Our approach is divided into three steps: candidate generating and weighting relying on an adaptive edit-distance, candidate scoring using language model and candidate ranking based on a regression model. Details of each step are discussed in the following subsections.

3.1 Candidate Generating and Weighting Based on an Adaptive Edit-Distance (Step 1)

In the first step, we generate candidates based on the character candidate graph which can deal with run-on and split-word errors. We then score such candidates using a modified confusion probability.

Candidate Generating: A string can be generated from the other string by edit operations of three edition types (deletion, insertion, or substitution). Therefore, we create the character candidate graph based on a "seed" word (an OCR error) with three corresponding node types (deletion, insertion, or substitution). Then we use this graph to generate candidates by one or more edit operations. More specifically, if two characters are generated from one "seed" character, this is a deletion node; otherwise, if one character is generated from two adjacent "seed" characters, it is an insertion node. In case that one character is substituted by one "seed" character, we have a substitution node.

Training dataset reveals that insertion and deletion caused by two adjacent characters are more common than those caused by three or more adjacent characters, therefore in this paper, we limit to two adjacent characters.

After graph construction, Breadth First Search (BFS) with some heuristic tuned from training dataset (maximum length of candidates, minimum confusion probability) is used to deal with the complexity.

The example graph is shown in Fig. 1. If "ar.d" is an OCR error, all "seed" characters 'a', 'r', '.', and 'd' are denoted as yellow nodes. High frequency substitution characters of 'a', 'r', '.', and 'd' are 'e', 'n', ',' and 'l', respectively, which are denoted as green nodes. Two adjacent characters 'r', '.' can be combined to generate the character insertion node 'n' denoted as a red node; one "seed" character 'd' can be divided into two characters "il" denoted as a blue node. One possible candidate of the error "ar.d" in Fig. 1 is "and" which is generated from substitution node 'a', insertion node 'n', and substitution node 'd'.

By using the character candidate graph, our approach can deal with two difficult error types, which are split-word errors (for instance, "appointed" is recognized as "ap pointed") and run-on errors (for example, "doubtfull of" is recognized as "doubtfud. of"). However, there are some limitations in quality control of too many candidates generated by one run-on error. Therefore, this paper only allows a punctuation and a digit be substituted by the space.

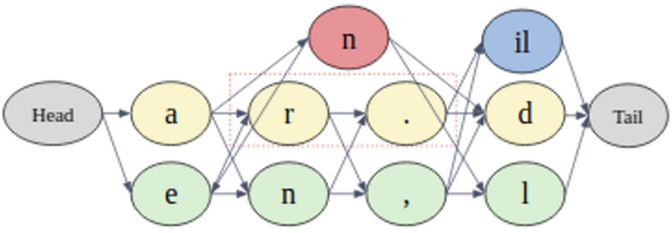

Fig. 1. Example of character candidate graph (Color figure online)

Candidate Weighting: The conditional probability $pr(x|w)$ of the given source word w recognized by the OCR software as the string x (also known as confusion probability of source word and OCR string) can be estimated by the confusion probabilities of the characters in x assuming that character recognition in OCR is an independent process [20].

Let $x_{1,i}$ be the first i characters of OCR string x and let $w_{1,j}$ be the first j characters of source string w. We define $pr(x_{1,i}|w_{1,j})$ to be the conditional probability that the substring $w_{1,j}$ is recognized as substring $x_{1,i}$ by the OCR process. $pr(x_{1,i}|w_{1,j})$ can be calculated as below:

$$pr(x_{1,i}|w_{1,j}) = \max \begin{cases} pr(x_{1,i}|w_{1,j-1}) * pr(del(w_j)) \\ pr(x_{1,i-1}|w_{1,j}) * pr(ins(x_i)) \\ pr(x_{1,i-1}|w_{1,j-1}) * pr(sub(x_i|w_j)) \end{cases} \quad (1)$$

In typical formula, the insertion, deletion conditioned on the previous character are computed as follows [5].

$$pr(del(w_j)) = \frac{del[w_{j-1}w_j]}{count[w_{j-1}w_j]}, \text{ if deletion} \quad (2)$$

$$pr(ins(x_i)) = \frac{ins[w_{j-1}x_i]}{count[w_{j-1}]}, \text{ if insertion} \quad (3)$$

$$pr(sub(x_1|w_j)) = \frac{sub[x_i, w_j]}{count[w_j]}, \text{ if substitution} \quad (4)$$

where $del[w_{j-1}w_j]$ is a number of times that the source characters $w_{j-1}w_j$ were recognized as w_{j-1} in the training set; $ins[w_{j-1}x_i]$ is a number of times that w_{j-1} was recognized as $w_{j-1}x_i$; $sub[x_i, w_j]$ is a number of times that w_j was recognized as x_i. These equations reveal that in order to calculate the confusion probability of insertion and deletion, there must be the same previous character.

Because erroneous OCR characters frequently appear together, two or more error characters can be recognized as one different correct character, or one character can be recognized as different correct characters [11]. It means that the insertion, deletion can depend on the different previous character instead of the same previous one. For example, "li" can be wrongly recognized as 'h', 'm' can be wrongly recognized as 'in', etc.

As a result, we propose to calculate the probability of deletion and insertion by using the probability of substitution of many characters by one character or one character by many characters.

$$pr(del(w_j)) = pr(sub(x_{i-1}|w_{j-1}w_j)) = \frac{sub[x_{i-1}, w_{j-1}w_j]}{count[w_{j-1}w_j]}, \text{ if deletion} \quad (5)$$

$$pr(ins(x_i)) = pr\big(sub\big(x_{i-1}x_i|w_{j-1}\big)\big) = \frac{sub\big[x_{i-1}x_i, w_{j-1}\big]}{count\big[w_{j-1}\big]}, \quad \text{if insertion} \tag{6}$$

For instance, the correction is "and", and the error is "ar.d". In this case, the correct letter 'n' is recognized as the error characters "r.". It is similar to insertion error type except that it does not have the same previous character, so we cannot apply the typical formula of insertion Eq. 3 directly. Typical approach (denoted as typical-prob. in experiments) uses the substitution formula twice to calculate that confusion probability:

$$\begin{aligned} pr(\text{`ar.d'}|\text{`and'}) = pr(sub(\text{`a'}|\text{`a'})) * pr(sub(\text{`r'}|\text{`n'})) \\ * pr(sub(\text{`.'}|\text{``})) * pr(sub(\text{`d'}|\text{`d'})) \end{aligned}$$

Our approach (denoted as modified-prob. in experiments) applies the substitution formula once:

$$pr(\text{`ar.d'}|\text{`and'}) = pr(sub(\text{`a'}|\text{`a'})) * pr(sub(\text{`r.'}|\text{`n'})) * pr(sub(\text{`d'}|\text{`d'}))$$

Our proposal also affects on creating the character confusion matrix. In particular, if one character (c) is recognized as two characters which are different from character (c), it is insertion. If two characters $(c1, c2)$ as recognized as one character which is different from characters $(c1, c2)$, it is deletion. Otherwise, it is substitution.

3.2 Candidate Scoring Using Language Model (Step 2)

After generating and weighting candidates at character level, in the second step, we utilize context information to score candidates of each OCR error.

Similar to some approaches of the context-based type, we consider the typical statistical language model (SLM) to deal with this problem. Moreover, we also explore the state-of-the-art recurrent neural network based language model (RNN-LM) [16] to compare two types of language models in context of erroneous OCR-ed text. SLM and LSTM are trained on the same training dataset used in "One Billion Word Language Model Benchmark" of Chelba *et al.* [3]. Each candidate is weighted according to the probability of trigram in SLM or that of predicting next word in RNN-LM.

In terms of SLM, the weight of each candidate is a sum of probabilities of three trigrams related to that candidate. For example, we have a phrase "yield to thbse who are", and two candidates {"those", "there"}, of the error "thbse". The weight of each candidate is calculated as follows:

$$\begin{aligned} weight_1(\text{``those''}) &= pr(\text{``yield to those''}) + pr(\text{``to those who''}) \\ &\quad + pr(\text{``those who are''}) \\ weight_1(\text{``there''}) &= pr(\text{``yield to there''}) + pr(\text{``to there who''}) \\ &\quad + pr(\text{``there who are''}) \end{aligned}$$

For constructing RNN-LM, we apply one of the most common type of RNN - Long Short Term Memory (LSTM) [8]. The weight of each candidate is a sum of probabilities of predicting the next word related to that candidate. More specifically, LSTM needs a seed which is a context to predict a next word. The candidate can appear in the context (seed) or be the next word.

To compare with trigram language model, we keep the total length of the seed and the next word to be three. For instance, we have the same phrase with SLM "yield to thbse who are" with the same error candidate {"those", "there"} of the error "thbse", the weight of each candidate is computed as below:

$$\begin{aligned} weight_2(\text{“those”}) &= pr(seed = \text{“yield to”}, next - word = \text{“those”}) \\ &+ pr(seed = \text{“to those”}, next - word = \text{“who”}) \\ &+ pr(seed = \text{“those who”}, next - word = \text{“are”}) \\ weight_2(\text{“there”}) &= pr(seed = \text{“yield to”}, next - word = \text{“there”}) \\ &+ pr(seed = \text{“to there”}, next - word = \text{“who”}) \\ &+ pr(seed = \text{“there who”}, next - word = \text{“are”}) \end{aligned}$$

3.3 Candidate Ranking Based on a Regression Model (Step 3)

After generating candidates and weighting them at character level and word level in two previous steps, this step reuses these features and some complementary features to predict the confidence of each candidate becoming a correction by a regression model. Then such confidence is used for candidate ranking. This step consists of two parts: feature extraction and candidate ranking.

Feature Extraction. Four important features used in our approach are selected and modified from a set of features of two related works [12, 15]. The first feature is "Probability of 3-length sequences related to errors" is the modified version of context feature, suggested by both of the related works; in terms of [12], this feature is "backward/forward bigram frequency"; in terms of [15], this is "exact/relax-context popularity". The second feature is "Probability of n-gram candidate", which is the general version of "unigram frequency" of two related works. Two last important features are features missing from each related work.

As analyzed in Sect. 2, "similarity feature" is an important feature used in real-word correction, which is ignored by [12]. Similarly, "confusion probability" is successfully used in several post-processing approaches, but is ignored by [15]. As to other features, we cannot use them because of different reasons. In fact, due to lack of information from the dataset, "word confidence" is ignored. We also remove the feature which is a part of other features, for example, "lexicon existence", which is included in "unigram frequency" feature. In addition, we refuse features that easily lead to bias such as "term frequency in OCR-ed text".

Let w_c be a candidate, C be a set of all candidates, and w_e be an OCR error. The details of each feature score are described in this section.

Probability of 3-Length Sequences Related to Errors: This feature is the normalized weight of step 2 mentioned in Sect. 3.2.

$$score(w_c, w_e) = \frac{weight_2(w_c)}{\sum_{w'_c \in C} weight_2(w'_c)} \tag{7}$$

Probability of n-Gram Candidate: Candidate can be a single word or a sequence of multiple words, it means that candidate is word n-gram. Instead of using the frequency of candidate and accepting 0 value if candidate is not in the training data, we use the probability of candidate in word n-gram model which already applied smoothing techniques for solving sparsity problem. This feature is the normalized probability of candidate in word n-gram model.

$$score(w_c, w_e) = \frac{pr(w_c)}{\sum_{w'_c \in C} pr(w'_c)} \tag{8}$$

Longest Common Subsequence (LCS) is an alternative in qualifying the similarity between two strings. Islam *et al.* [9] proposed two variations of LCS, including Normalized Longest Common Subsequence (NLCS) and Normalized Maximal Consecutive Longest Common Subsequence (NMCLCS).

NLCS considers the length of both the shorter and the longer string for normalization, as follows:

$$NLCS(w_c, w_e) = \frac{2 * len(LCS(w_c, w_e))^2}{len(w_c) + len(w_e)} \tag{9}$$

MCLCS requires the consecutive common subsequence. There are three variations of MCLCS with additional conditions. MCLCS_1 and MCLCS_n use substrings beginning at the first and the n-th character, respectively; MCLCS_z only considers substrings ending at the last character.

$$NMCLCS_i(w_c, w_e) = \frac{2 * len(MCLCS_i(w_c, w_e))^2}{len(w_c) + len(w_e)} \tag{10}$$

where MCLCS_i can be MCLCS_1, MCLCS_n or MCLCS_z.

Confusion Probability: This feature is the normalized weight of step 1 in Sect. 3.1:

$$score(w_c, w_e) = \frac{weight_1(w_c)}{\sum_{w'_c \in C} weight_1(w'_c)} \tag{11}$$

Candidate Ranking. A regression model is used for scoring candidates. For training and testing a regressor, if a candidate is a correction, its feature vector is labeled as 1. Otherwise, the feature vector is labeled as 0. Candidate with the highest confidence is suggested as the correction.

However, correcting run-on errors often produces irrelevant candidates which cause a big difference between corrected word and ground truth one. For example, post-processing tries to find the most suitable candidate from a dictionary for a run-on error "where loan", and suggests the top candidate such as "helios" which is totally different from the GT "where I can". Therefore, the final filter based on the edit distance between error and its top candidate decides whether use the top candidate or keep the error.

4 Experiments

In this section, we first describe details of the dataset and metric used in the evaluation. Then we analyze the performance of our approach in comparison with the results of the teams taking part in the competition.

4.1 Evaluation Dataset

The English monograph dataset of the ICDAR 2017 Post-OCR text correction competition [4] with 666 training documents and 41 testing documents is used for evaluating our approach. These documents are selected from digital collections of the National Library of France (BnF) and the British Library (BL). The corresponding GT has been extracted from BnF's internal projects and can be obtained through the competition's homepage[1].

4.2 Evaluation Metric

We use the average Levenshtein distances as the evaluation metric, same as used in the Post-OCR text correction competition ICDAR 2017 [4]. The metric can be viewed as the modified version of character error rate, which considers the confidence of each candidate to be the correction in case that there are many candidates of one error.

$$avgDistance = \frac{\sum_{i=1}^{n} weight_i * distance(candidate_i, correctWord_i)}{N} \tag{12}$$

where $weight_i$ is the confidence of $candidate_i$ to be the correct word, n is a number of OCR errors, and N is a total number of characters in reference.

This official metric of the competition is intended to take into account partial improvement, even when the word correction is not fully matching the GT.

The improvement percentage is calculated based on the comparison of the original distance (the distance between the OCR output and GT) and the corrected distance (the distance between the corrected output and GT).

[1] https://sites.google.com/view/icdar2017-postcorrectionocr/, last visited on 28 June 2018.

4.3 Evaluation Results

By combining two ways of calculating step 1's weight and two ways of calculating step 2's weight, we consider four approaches in total: "typical-prob.SLM", "modified-prob. SLM", "typical-prob.LSTM" and "modified-prob.LSTM". The overall performance of our approaches is shown in Table 1.

In terms of performance of step 1, as mentioned in Sect. 3.1, our approach of calculating the confusion probability considered the common way in which erroneous characters appear, therefore our suggestion "modified-prob." strongly outperforms the typical approach "typical-prob." even with different techniques of step 2 (SLM or LSTM).

As to the performance of step 2, in erroneous OCR-ed context, SLM has slightly better performance than LSTM, with about 1.8% of relative increase of improvement in case of "typical-prob." and 2.4% in case of "modified-prob".

Regarding the performance of step 3, experimental results show that gradient boosting regression model [7] on top of decision trees with the least square loss function outperforms other regression models.

Among 10 participants of the correction task in the Post-OCR text correction ICDAR 2017, only six teams can improve the distance with GT, and other teams almost deteriorate documents. These final results confirmed the difficulty of post-processing step to deal with the noisy OCR output and the uncleaned GT.

Our best regression model ("modified-prob.SLM") obtains higher performance than 9 teams. It should be emphasized that our approach is better than the multi-modular approach of statistical machine translation approach and spelling checker (MMDT) and the neural machine translation approach applied on post-OCR problem (CLAM), with around 51% and 4.1% relative increase, respectively. Although our result is still lower than the one of the best-performing participant (Char-SMT/NMT), it is unfair to compare our multi-modular approach with the ensemble one. While our solution uses only one best regression model to score candidates, Char-SMT/NMT trained several models of both statistical and neural machine translation, and then combined the top candidates generated from such models.

Table 1. Overall performance scores over English monograph dataset of the Post-OCR text correction ICDAR 2017 with top 4 competition teams.

Approach	Improvement (%)
Baselines	
Char-SMT/NMT	**43**
CLAM	29
WFST-PostOCR	28
MMDT [18]	20
Proposed methods	
modified-prob.SLM	**30.2**
modified-prob.LSTM	29.5
typical-prob.SLM	22.4
typical-prob.LSTM	22

In addition, if we recommend top 3, top 6 candidates for each error, the best improvement percentage of our best approach "modified-prob.SLM" is 40.5% and 42.8%, respectively. It should be clear that the best improvement percentage is calculated based on the original distance and the best corrected distance (the distance of the most relevant candidate among the top n candidates with the GT word). In other words, in semi-automatic mode, our multi-modular approach can suggest correct candidates with the comparable performance with the ensemble approach of the winner Char-SMT/NMT.

5 Conclusion

In this paper, we explored a modified method of generating and calculating the confusion probability, which has better performance than the standard edit distance method. When we compare the performance of SLM and LSTM in erroneous OCR-ed contexts, the experiments show that SLM is slightly better than LSTM in terms of 3-length sequence. Finally, by extracting important features suggested in two previous works and using a regression model for candidate ranking, our best approach is comparative with approaches in ICDAR2017 Competition on Post-OCR Text Correction though it still under-performs the ensemble approach of machine translation techniques.

References

1. Afli, H., Barrault, L., Schwenk, H.: OCR error correction using statistical machine translation. Int. J. Comput. Linguist. Appl. **7**, 175–191 (2016)
2. Bassil, Y., Alwani, M.: OCR post-processing error correction algorithm using Google online spelling suggestion. arXiv preprint arXiv:1204.0191 (2012)
3. Chelba, C., et al.: One billion word benchmark for measuring progress in statistical language modeling (2013)
4. Chiron, G., Doucet, A., Coustaty, M., Moreux, J.P.: ICDAR2017 competition on post-OCR text correction. In: 2017 14th IAPR International Conference on Document Analysis and Recognition, ICDAR, vol. 1, pp. 1423–1428. IEEE (2017)
5. Church, K.W., Gale, W.A.: Probability scoring for spelling correction. Stat. Comput. **1**(2), 93–103 (1991)
6. Evershed, J., Fitch, K.: Correcting noisy OCR: context beats confusion. In: Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage, pp. 45–51. ACM (2014)
7. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Ann. Stat. 1189–1232 (2001)
8. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
9. Islam, A., Inkpen, D.: Semantic text similarity using corpus-based word similarity and string similarity. ACM Trans. Knowl. Discov. Data **2**, 10 (2008)
10. Islam, A., Inkpen, D.: Real-word spelling correction using Google Web IT 3-grams. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, vol. 3, pp. 1241–1249 (2009)

11. Jones, M.A., Story, G.A., Ballard, B.W.: Integrating multiple knowledge sources in a Bayesian OCR post-processor. In: International Journal on Document Analysis and Recognition, p. 925–933 (1991)
12. Kissos, I., Dershowitz, N.: OCR error correction using character correction and feature-based word classification. In: 2016 12th IAPR Workshop on Document Analysis Systems, DAS, pp. 198–203. IEEE (2016)
13. Koehn, P., et al.: Moses: open source toolkit for statistical machine translation (2007)
14. Llobet, R., Navarro-Cerdan, J.R., Perez-Cortes, J.C., Arlandis, J.: Efficient OCR post-processing combining language, hypothesis and error models. In: Hancock, E.R., Wilson, R. C., Windeatt, T., Ulusoy, I., Escolano, F. (eds.) SSPR/SPR 2010. LNCS, vol. 6218, pp. 728–737. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14980-1_72
15. Mei, J., Islam, A., Wu, Y., Moh'd, A., Milios, E.E.: Statistical learning for OCR text correction. arXiv preprint arXiv:1611.06950 (2016)
16. Mikolov, T., Karafiát, M., Burget, L., Černocký, J., Khudanpur, S.: Recurrent neural network based language model. In: Eleventh Annual Conference of the International Speech Communication Association (2010)
17. Niwa, H., Kayashima, K.: Postprocessing for character recognition using keyword information
18. Schulz, S., Kuhn, J.: Multi-modular domain-tailored OCR post-correction. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 2716–2726 (2017)
19. Tiedemann, J.: Character-based pivot translation for under-resourced languages and domains. In: Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, pp. 141–151 (2012)
20. Tong, X., Evans, D.A.: A statistical approach to automatic OCR error correction in context. In: Fourth Workshop on Very Large Corpora (1996)