OCR Post-Processing Text Correction using Simulated Annealing (OPTeCA)

Gitansh Khirbat
Computing and Information Systems
The University of Melbourne
Australia
gitansh.khirbat@unimelb.edu.au

Abstract

This paper describes the system details and results of team "EOF" from the University of Melbourne for the shared task of ALTA 2017, which addresses the problem of text correction for post-processed Optical Character Recognition (OCR) based systems. We developed a two stage system which first detects errors in the given OCR post-processed text with the help of a support vector machine trained using given training dataset, followed by rectifying the errors by employing a confidence-based mechanism using simulated annealing to obtain an optimal correction from a pool of candidate corrections. Our system achieved a F_1-score of 32.98% on the private leaderboard[1], which is the best score among all the participating systems.

1 Introduction

The dawn of digital age on mankind has laid the foundation of connectivity, fostering access and exchange of information practically anywhere in the world. Information can be present in any form, the most common being textual and graphical documents. Capturing and curating documents such as magazines, newspapers, journals and scientific articles is the primary requirement for a digitized, inter-connected society. While most of the textual documents can be stored with a decipherable textual component, the story is not the same for graphical documents which can be a collection of images containing scans of a textual document.

Optical Character Recognition (OCR) is the process of identifying typed, handwritten or printed textual characters within a document containing scanned images or photographs with the help of various image processing and pattern recognition techniques (Tappert et al., 1990), (Gupta et al., 2007). The text obtained by OCR systems often suffers from low accuracy owing to irregularities in images, poor scans or simply the nature of arrangement of letters in a word. For example, reading "lwo" instead of "two", "ia" instead of "is", "m" instead of "rn", to name a few. These erroneous characters severely hamper the quality and readability of a converted document. Identifying and rectifying such erroneous characters in every OCR-processed document manually is a tedious task due to the sheer volume of data. Consequently, a methodology is required to identify such OCR errors and rectify them in order to enforce standards of purity and quality of the archived data. This need has motivated the shared task of ALTA 2017 (Molla and Cassidy, 2017). The task organizers have provided the original outputs of an OCR system together with their corrected version for scanned Australian publications from Trove database[2]. Using this data, the participants are asked to automatically identify and rectify the OCR errors for documents in a separate test dataset.

Considerable research has been conducted previously to automatically correct text obtained by OCR systems using machine learning. (Lund and Ringger, 2009) (Lund et al., 2011) (William B. Lund, 2013) (Lund et al., 2014) introduced various techniques to select the most appropriate correction among a pool of candidates. (Jones and Eisner, 1992), (Kukich, 1992) demonstrated that OCR-generated errors are more diverse than handwriting errors. (Taghva and Stofsky, 2001) used extensive feature engineering to facilitate a robust candidate selection using a probabilistic model. Motivated by (Mei et al., 2016), we adopt a two stage approach to solve this task. First, our system detects errors in the given OCR post-processed

[1] https://www.kaggle.com/c/alta-2017-challenge/leaderboard

[2] http://trove.nla.gov.au

Gitansh Khirbat. 2017. OCR Post-Processing Text Correction using Simulated Annealing (OPTeCA). In *Proceedings of Australasian Language Technology Association Workshop*, pages 119−123.

text with the help of a support vector machine (SVM) trained using given training dataset. This is followed by identifying a set of candidate words as corrections for each of the errors guided by allowing a limited number of character modifications. Finally, we rank the candidates by employing a confidence based mechanism using simulated annealing to obtain an optimal correction from the set of candidate corrections.

The rest of the paper is organized as follows. Section 2 describes the methodology in detail. Section 3 describes the experiments and results. Section 4 discusses the error analysis of the obtained results and Section 5 concludes the paper.

2 Methodology

OCR post-processing text correction is a challenging and complex problem. The ever-growing vocabulary constrains it further. In order to solve this problem, we break it down into two sub-problems, namely, identification of erroneous terms from the post-processed OCR text, followed by rectification of the identified erroneous terms. The complete pipeline is shown in Figure 1, with the explanation of each stage as follows.

2.1 Error detection

The first stage of our system is to detect the erroneous terms for a given document. It involves two components as described below.

2.1.1 Pre-processing

The pre-processing module consists of tokenization of a given textual document, i.e. the original output of the OCR. We defined regular expression patterns which split a textual document on delimiters such as full-stop (.), comma (,), semi-colon (;), single quotes (‘,’) or double quotes (“,”). The tokens are considered for further processing *as-is*, i.e. without undergoing lemmatization. The primary reason to abstain from lemmatization is to preserve the original OCR words in order to rule out the scope of any character-based discrepancy. Additionally, care is taken to preserve the token order. The order is important as it dictates one of the features as defined in Section 2.1.2.

2.1.2 Feature Extraction and Classification

The next step is to classify each token in the document as being erroneous or free of any error. We train a SVM with radial basis function (RBF) kernel and made use of the following features:

- Presence of non alpha-numeric text within a word is one the strongest indicators of an erroneous word. These mainly include special symbols like ‘$’, ‘#’, ‘%’ and punctuation marks like ‘!’, ‘?’, ’;’, ’:’, etc. For example, “th?” and “Mr. Pat?rsom” contain a punctuation mark ‘?’; “***n”, “JM**shopB” contain a special symbol ‘*’. We created a dictionary of such special symbols as observed from the given training data.

- The bigram frequency of a word should be greater than a frequency threshold that varies with different word length. A common word is less likely to be an error word. We adopted this feature from (Mei et al., 2016).

- A word is likely to be correct if this word with its context occurs in other places. We use a sliding window similar to (Mei et al., 2016) to construct n-gram contexts for a word. The frequency of one of the context in n-gram corpus should be greater than a frequency threshold.

Using these features, we train a binary SVM classifier (Pedregosa et al., 2011) which classifies a word being erroneous (1) or not (0). The experimental details are mentioned in Section 3.2.

2.2 Error rectification

The second stage of our system solves the problem of rectifying the erroneous words identified in the first stage. It consists of two major components, namely candidate search and candidate ranking as described below.

2.2.1 Candidate Search

In this module, for each erroneous word, a set of candidate corrections is recommended within a limited number of character modifications based on calculating minimum edit distance between the erroneous word and the candidate correction. We make use of Levenshtein's edit distance (Levenshtein, 1966) to calculate the minimum edit distance consisting of the standard three operations, namely, insertion, deletion or substitution. The threshold is chosen heuristically on the basis of experiments conducted.

2.2.2 Candidate Ranking

This module makes use of the output of previous module, i.e. a set of candidate corrections (w_{ci},

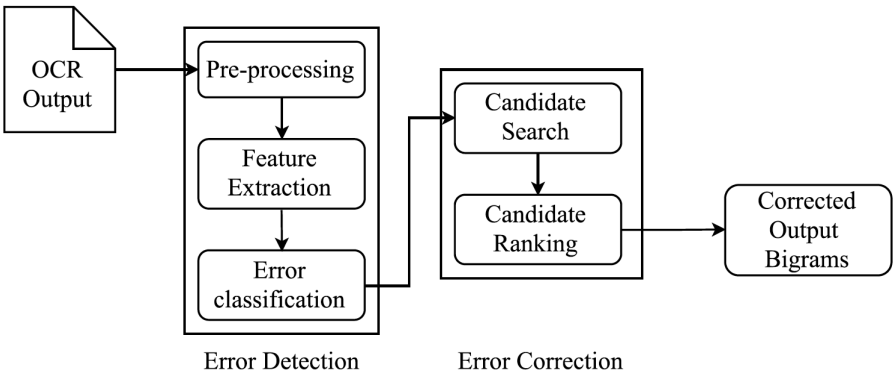

Figure 1: System pipeline

$i \in \mathbb{W}$) for each erroneous term (w_e), to assign a score to each candidate correction using simulated annealing (SA) algorithm (Kirkpatrick et al., 1983). SA requires an aperiodic Markov chain defined on a certain state space, and a cooling schedule to iteratively push the solution towards the optimum. In this module, the state space is set of all candidate corrections. We calculate a similarity score for each of the candidate corrections (w_{ci}) on the basis of the following three factors:

1. Minimum edit distance $d(w_{ci}, w_e)$ as calculated by Levenshtein's edit distance.

2. Normalized longest common subsequence (Allison and Dix, 1986) which takes into account the length of both the shorter and the longer string for normalization.

$$nlcs(w_{ci}, w_e) = \frac{2 * len(lcs(w_{ci}, w_e))^2}{len(w_{ci}) + len(w_e)} \quad (1)$$

3. Normalized maximal consecutive longest common subsequence, which is a modification of aforementioned factor by limiting the common subsequences to be consecutive.

$$nmnlcs(w_{ci}, w_e) = \frac{2 * len(mclcs(w_{ci}, w_e))^2}{len(w_{ci}) + len(w_e)} \quad (2)$$

The final score is calculated as a weighted sum of these three factors:

$score(w_{ci}, w_e) = \alpha_1 * d(w_{ci}, w_e) + \alpha_2 * nlcs(w_{ci}, w_e) + \alpha_3 * nmnlcs(w_{ci}, w_e)$

where, α_1, α_2 and α_3 are chosen heuristically. Next, we perturb the given candidate 26 times, i.e. for all the characters of English alphabet, in order to check which character returns the maximal score. This is followed by validating the presence of that candidate correction by Google Web n-gram corpus[3]. Finally, the candidates are ranked on the basis of this optimized score. The candidate having highest score is returned as the final suggested correction.

3 Experiments and Results

The ALTA shared task is to rectify textual errors in OCR post-processed documents. We first describe the given dataset briefly, followed by experimental setup and results.

3.1 Dataset

The shared task organizers obtained a corpus of approximately 8,000 Australian publications from Trove database. The corpus consists of original output of OCR system for each of the documents, along with their corrected versions. The organizers have provided 6,000 documents and their corrected versions as training dataset. 1,941 documents are provided as the test dataset, for which only the original output of the OCR system is provided. The details of data are given by (Molla and Cassidy, 2017).

3.2 Experimental Setup and Results

Stage 1 of our experiment pertains to error detection which classifies each word of the document to be either erroneous (1) or correct (0). In order to train a binary SVM classifier, we split the given training data into training and development datasets using 5-fold cross validation. The RBF kernel is employed to train the SVM. The test

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

Features	P	R	F1
Non-alphanumeric character presence	63.2	42.1	50.5
+bigram frequency	67.3	43.9	53.1
+n-gram contexts	69.6	44.2	54.1

Table 1: Stage 1 - Error detection using SVM

Model	$\mathbf{F1}_{public}$	$\mathbf{F1}_{private}$
Baseline	22.86	23.09
$SA_{0.80}$	25.44	25.83
$SA_{0.85}$	27.52	28.05
$SA_{0.88}$	29.71	30.14
$SA_{0.92}$	**32.98**	**33.48**

Table 2: Stage 2 results - Error rectification

dataset remains unused since the correct labels for erroneous words are unknown. Table 1 reports the intermediate results obtained by adding the features defined in Section 2.1.2 incrementally.

For the stage 2 subproblem of error rectification, first we select the threshold for the Levenshtein's edit distance by measuring the minimum edit distance between the words obtained from corrected and original documents provided in the training dataset. This helps in recommending the candidate corrections by allowing words for which minimum edit distance is less than or equal to the threshold. For candidate ranking, we initialize the score as 0 for each pair of (w_{ci}, w_e) corresponding to an erroneous word. The value of temperature is initialized to 500 and cooling schedule is initialized to 0.8. Table 2 shows the five models that were used to render final results. The baseline model corresponds to ranking of candidate corrections on the basis of total score calculated in Section 2.2.2. $SA_{0.80}$, $SA_{0.85}$, $SA_{0.88}$ and $SA_{0.92}$ correspond to models trained using simulated annealing at the respective cooling schedules.

The trained model is used for predictions corresponding to the public leaderboard which contains 50% of the total data. Finally, at the end of the competition, the predictions are measured against the remaining 50% of data which corresponds to the private leaderboard. The results obtained by using the aforementioned features is shown in Table 2. Standard precision, recall and F1-score metrics are used to report the prediction results.

4 Discussion

Our system performs almost similarly on both public and private leaderboards, which indicates that the model is not overfitting. Table 1 indicates that a collective use of character-level features and contextual features leads to an increase in F1 score, even if it's a marginal increment. The recall of our error detection module is consistently low, which demonstrates the complexity of this sub-problem. Table 2 demonstrates that simulated annealing has proven to show an improvement of about 5% F_1 score over the baseline score-based model.

What worked well: Our system was able to rectify some of the punctuation based errors like "Collision_;" → "Collision_<next-word>". We were also able to rectify certain typo-based errors like "ofi" → "of".

What did not work: Our system does not always return a correction when text containing a number is identified as an erroneous term. For example, in "October 2fi", the term "2fi" remains undetected.

There are many other features which we could have tried like considering erroneous text location in the document, syntactic structure of sentences within the document and non-English text words, to name a few. However, given the limitation of time, it was not possible to incorporate these features. It would be interesting to expand this system by adding these features in future.

5 Conclusion

OCR post-processed text correction is an important and challenging problem that needs to be addressed to facilitate digitization. In this paper, we describe our participating system, which was based on a supervised classification method to detect erroneous words, followed by suggesting optimal corrections for each erroneous word with a confidence-based mechanism using simulated annealing. Our system was ranked the best with an F1-score of 32.98%.

References

Lloyd Allison and Trevor I Dix. 1986. A bit-string longest-common-subsequence algorithm. *Information Processing Letters*, 23(5):305–310.

Maya R. Gupta, Nathaniel P. Jacobson, and Eric K. Garcia. 2007. Ocr binarization and image pre-

processing for searching historical documents. *Pattern Recogn.*, 40(2):389–397, February.

Mark A Jones and Jason M Eisner. 1992. A probabilistic parser and its applications. In *AAAI Workshop on Statistically-Based NLP Techniques*, pages 20–27.

Scott Kirkpatrick, C Daniel Gelatt, Mario P Vecchi, et al. 1983. Optimization by simulated annealing. *science*, 220(4598):671–680.

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

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

William B. Lund and Eric K. Ringger. 2009. Improving optical character recognition through efficient multiple system alignment. In *Proceedings of the 9th ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '09, pages 231–240, New York, NY, USA. ACM.

W. B. Lund, D. D. Walker, and E. K. Ringger. 2011. Progressive alignment and discriminative error correction for multiple ocr engines. In *2011 International Conference on Document Analysis and Recognition*, pages 764–768, Sept.

William B Lund, Eric K Ringger, and Daniel D Walker. 2014. How well does multiple ocr error correction generalize? Society of Photo-Optical Instrumentation Engineers.

Jie Mei, Aminul Islam, Yajing Wu, Abidalrahman Moh'd, and Evangelos E Milios. 2016. Statistical learning for ocr text correction. *arXiv preprint arXiv:1611.06950*.

Diego Molla and Steve Cassidy. 2017. Overview of the 2017 alta shared task: Correcting ocr errors. In *Proceedings of Australasian Language Technology Association Workshop*, Brisbane, Australia.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Kazem Taghva and Eric Stofsky. 2001. Ocrspell: an interactive spelling correction system for ocr errors in text. *International Journal on Document Analysis and Recognition*, 3(3):125–137.

C. C. Tappert, C. Y. Suen, and T. Wakahara. 1990. The state of the art in online handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(8):787–808, Aug.

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