A Novel Machine Learning Based Approach for Post-OCR Error Detection

Shafqat Mumtaz Virk
Språkbanken Text, Dept. of Swedish
University of Gothenburg
405 30 Gothenburg, Sweden
virk.shafqat@gmail.com

Dana Dannélls
Språkbanken Text, Dept. of Swedish
University of Gothenburg
405 30 Gothenburg, Sweden
dana.dannells@svenska.gu.se

Azam Sheikh Muhammad
Chalmers University of Technology
412 96 Gothenburg, Sweden
sheikhazam@gmail.com

Abstract

Post processing is the most conventional approach for correcting errors that are caused by Optical Character Recognition (OCR) systems. Two steps are usually taken to correct OCR errors: detection and corrections. For the first task, supervised machine learning methods have shown state-of-the-art performances. Previously proposed approaches have focused most prominently on combining lexical, contextual and statistical features for detecting errors. In this study, we report a novel system to error detection which is based merely on the n-gram counts of a candidate token. In addition to being simple and computationally less expensive, our proposed system beats previous systems reported in the ICDAR2019 competition on OCR-error detection with notable margins. We achieved state-of-the-art F1-scores for eight out of the ten involved European languages. The maximum improvement is for Spanish which improved from 0.69 to 0.90, and the minimum for Polish from 0.82 to 0.84.

1 Introduction

Post processing is the most conventional approach for correcting errors that are caused by Optical Character Recognition (OCR) systems. Traditionally, the task is divided into two subtasks: (1) Error detection to classify words[1] as either erroneous or valid, and (2) Error correction to find suitable candidates to correct the erroneous words (Kolak and Resnik, 2005; Kissos and Dershowitz, 2016; Mei et al., 2016). A large body of work has proven the success of statistical and supervised machine learning methods for both subtasks (Afli et al., 2016; Schulz and Kuhn, 2017; Nguyen et al., 2018; Amrhein and Clematide, 2018; Nguyen et al., 2019).

Machine learning methods largely rely on feature engineering for their performances, particularly in supervised settings. Feature engineering involves exploring various features and feature combinations that best characterise the data. However, for post-OCR error detection, finding a suitable set of features is challenging because of the diversity of OCR errors (Amrhein and Clematide, 2018). This has been demonstrated in previous work and more recently in the ICDAR competitions (Chiron et al., 2017; Rigaud et al., 2019) where various features have been explored with varying success rates (more details in Section 2).

To address this challenge we propose a novel approach to the error detection task. Instead of examining more features we focus merely on a single feature, namely the n-gram counts of the candidate token. Our approach is inspired by dictionary lookup approaches, which are known for their simplicity and efficiency but are restricted to the dictionary size. Since building large-scale dictionaries is a challenging task in itself, we propose to generate n-grams of a given candidate token, and then use their counts as the only feature to train machine learning models (more details in Section 3). We evaluate our method on the ICDAR2019 dataset (Chiron et al., 2017) and compare the results to a number of approaches reported in the competition (Section 4). Our system achieves state-of-the-art results for eight out of the ten involved languages by beating the previous results with fair margins and comparable scores for the remaining two languages. Our approach is very simple and is computationally less expensive as it does not require any other feature computation apart from the n-gram counts.[2]

[1]We write ‘word’ although, in practice, it actually refers to any token in our data – not necessarily a lexical word. We will use the term ’token’ and ’word’ alternatively throughout the paper.

[2]The data and models are available under CC BY

2 Related Work

Our work takes inspiration from dictionary based approaches and uses supervised machine learning techniques to train a post-OCR error detection model. We therefore focus on reviewing the works related to dictionary and statistical supervised machine learning based approaches.

Simple dictionary based approaches has performed reasonably well for various natural language processing tasks (e.g. Hull and Grefenstette (1996); Sindhu and Sagar (2017)). It is therefore not surprising the approach has been applied to detecting OCR errors before (Schulz and Kuhn, 2017; Nguyen et al., 2018). Taking this approach, dictionaries and large word lists are usually compiled from corpora and other sources. Each token in the data is then compared with the word in the dictionary to determine whether it is an error word or not. This approach has been explored by the CSIITJ team who was among the six successful teams participating in the ICDAR 2019 competition (Rigaud et al., 2019). The team complied a dictionary of over 370 thousand English and French words and checked each word in the dataset against it.

Dictionary lookup methods for post-OCR are challenging because they usually suffer from out-of-vocabulary problem. Another limitation is in detecting real-word errors, i.e. the word appears in the dictionary but is wrong in its context. Therefore alternative methods to OCR error detection have been proposed (see a comprehensive survey of existing methods by Nguyen et al. (2021)).

The remaining teams in ICDAR 2019 applied techniques from: (i) Context-based character correction using BERT (CCC) – the winning system; (ii) Character level attention approach using the open source system OpenNMT (CLAM); (iii) Weighted finite-state transducers based on noisy channel model (REA1&2); and (iv) Character level seq2seq multi-layer LSTM (UVA).

Other approaches to post-OCR error detection combined character-, word-n-grams and context based features to train a machine learning model (Mei et al., 2016; Khirbat, 2017; Nguyen et al., 2019; Dannélls and Persson, 2020). Mei et al. (2016) trained a regression model on 6 features containing n-gram and context information. Khirbat (2017) trained a support vector machine (SVM) model with 3 features: presence of non-alphanumeric characters, bi-gram frequency of the word and context information that is if the word appears with its context in other places. Nguyen et al. (2019) trained a Gradient Tree Boosting classifier on a set of 13 character and word features on two datasets of English historical handwritten documents (monograph and periodical) taken from the ICDAR competition (Chiron et al., 2017). The features they experimented with include character and word n-gram frequencies, part-of-speech, and the frequency of the OCR token. Dannélls and Persson (2020) trained an SVM model on 6 statistical and word based features including the number of non-alphanumeric characters, number of vowels, word length, tri-gram character frequencies, number of uppercase characters and the amount of numbers occurring in the word.

As many authors point out the choice of the features is essential for the performance of the machine learning model. The advantage of these methods is that they are trained to detect both real-word and non-word errors. The drawback is they require laborious feature engineering.

Laborious feature engineering is a bottleneck not only for machine learning but also for other statistical approaches that rely on pre-defined features extracted from data, such as noisy channel approaches (Evershed and Fitch, 2014; Kissos and Dershowitz, 2016; Drobac et al., 2017).

3 Method

3.1 Datasets and Preprocessing

We used the datasets from the ICDAR2019 competition on post OCR error detection and correction.[3] The total size of the original data is 22 million characters and it contains varying numbers of characters for ten European languages (Bulgarian, Czech, German, English, Spanish, Finnish, French, Dutch, Polish, Slovenian) together with the corresponding ground truth data. The dataset comes with the OCRed and ground truth aligned at the character level. For our experiments, we needed to align it at the token (word) level. We did that by tokenizing the ground truth at space and for each token taking the same number of characters from the OCRed version. After we removed the special alignment symbols ('@' and '#') inserted by organizers for alignment. The resulting OCRed and ground truth tokens were compared to set the labels '0' if the

4.0 licence at `https://github.com/spraakbanken/NovelOCRErrorCorrection`.

[3]https://sites.google.com/view/icdar2019-postcorrectionocr/dataset

token was erroneous or '1' if the token was valid. These labels are the dependent variables that are to be learned and predicted by the machine learning models. Table 1 shows example tokens, ground truth, and the labels from English, Danish, and Finnish datasets. Table 2 shows the number of training and test sets produced for each language after removing the duplicates and instances with 'NA' values and the class percentage i.e. what percentage of the total is the class '0' and '1' in the training and test sets.

3.2 Machine Learning Models and Encoding

Machine learning classifiers are known to have pros and cons depending on the task at hand. Dannélls and Virk (2020) compared between 5 state-of-the-art machine learning classifiers including Logistic Regression, Decision Tree, Bernoulli Naive Bayes, Naive Bayes and SVM. They found that SVM is the best choice for post-OCR detection. Others have also shown the performance of SVM is equivalent to the performance of artificial neural networks (Arora et al., 2010; Hamid and Sjarif, 2017; Amrhein and Clematide, 2018). In response to these previous experiments, in this study we have chosen to experiment with SVM models.

We take advantage of the implementation of the machine learning algorithms in the *sklearn* module (Pedregosa et al., 2011). Because the module requires the data to be in numeric form we used one-hot encoding for data transformations (see Section 4.1). While the details of the encoding method are beyond the scope of this paper, the major idea behind one-hot encoding is to add an extra dimension in the feature vector for each unique feature value. This produces an N dimensional feature vector (the learned encoding), where N is the total number of unique values of all features. In our case we have words as the training data. Suppose [w_1,w_2,w_3......w_n] is the set of unique words, and [0,1,1......0] is the set of corresponding labels representing whether the word is erroneous or not. The learned one-hot encoding will be a (n+2) dimensional vector, where *n* is the unique number of words in the training data and there are 2 unique label values. Each word is then encoded by setting the corresponding word and label dimensions of the vector to 1 while the remaining dimensions are set to 0.

3.3 Generating the Machine Learning Features

As mentioned previously, we took inspiration from dictionary look up based approaches, but in this study we have applied it in a novel way. Instead of building a dictionary separately from different external resources, we let our SVM model build it from the training data. This was achieved by learning one-hot encoding from the training data (i.e. words), encoding the training data and then using the resulting vectors as the only feature to train the SVM model. In other words, we turn the model into a dictionary lookup kind of system as the model memorizes vectors of each training instance. During prediction the trained model simply relies on the observed word, i.e. whether its encoded feature vector has been seen in the training data and predicts accordingly.

This type of approach has a major restriction that it is not scalable and is bound to feature values seen in the training data. In our case this means if a word has not been seen in the training data the system will simply fail to predict whether it is erroneous or not. Another downfall is that depending on the size of training data it may take days to train such models. To overcome these limitations we experimented further with the n-gram approach. Instead of using the complete word, for each candidate token, we generated character uni-grams, bi-grams, and tri-grams from it. These n-grams together with their counts within the token were used as feature values to train and test the model. To take an example, suppose our candidate word is 'passenger'. The computed uni-, bi-, and tri-gram counts will be as follows:

- **uni-gram** {'a':1, 'e':2, 'g':1, 'n':1, 'p':1, 'r':1, 's':2}
- **bi-gram** {' p':1, 'as':1, 'en':1, 'er':1, 'ge':1, 'ng':1, 'pa':1, 'r ':1, 'se':1, 'ss':1}
- **tri-gram** {' pa':1, 'ass':1, 'eng':1, 'er ':1, 'ger':1, 'nge':1, 'pas':1, 'sen':1, 'sse':1}

It is worth mentioning that the scope of n-gram counts is limited to the word itself, rather than the entire training data i.e. these counts represent the occurrence of a particular uni-, bi-, or tri-gram within the word rather than the total count of the n-gram in the training data. Previous authors have attempted to exploit n-gram frequencies (e.g. Mei et al. (2016); Khirbat (2017)) computed over the

English			Danish			Finnish		
Word	GT	Label	Word	GT	Label	Word	GT	Label
matter	matter	1	Bezirke	Bezirke	1	jolloin	jolloin	1
the	the	1	.Fiili	@F@li	0	lainasimat	lainasiwat	0
king@	king	0	welche	welche	1	saimat	saiwat	0
very	very	1	niedergese@ht	niedergesetzt	0	takaisin	takaisin	1
glad	glad	1	Bericht	Bericht	1	jtt	jtt	1
hereof,@	hereof,	0	Wesentlichen	Wesentlichen	1	kaupungissa	kaupungissa	1
@Hkewise	likewise	0	hkle	hatte	0	Maan	Waan	0

Table 1: A sample from the English, Danish, and Finnish datasets after the preprocessing step (GT = Ground Truth).

Language	Training Set				Test Set			
	0		1		0		1	
	#words	%	#words	%	#words	%	#words	%
Bulgarian (BG)	17844	0.37	29635	0.63	9750	0.40	14404	0.60
Czech (CZ)	7227	0.19	31230	0.81	3335	0.27	9081	0.73
Danish (DE)	18216	0.29	45325	0.71	3573	0.25	10784	0.75
English (EN)	9844	0.33	20559	0.67	3802	0.33	7609	0.67
Spanish (ES)	33164	0.51	31698	0.49	8996	0.59	6287	0.41
Finnish (FI)	48644	0.22	165940	0.78	11996	0.23	39781	0.77
French (FR)	85678	0.20	343858	0.80	21855	0.20	85535	0.80
Dutch (NL)	45593	0.51	47875	0.49	15619	0.47	17712	0.53
Polish (PL)	21436	0.70	8912	0.30	6730	0.57	5008	0.43
Slovenian (SL)	6098	0.16	31865	0.84	5128	0.31	11501	0.69

Table 2: Training and test dataset statistics

entire training data, but not in the sense we are proposing in this study. This is what makes our approach novel. The intuition for using n-grams instead of complete word to overcome the above mentioned limitation is simple: For a given word, it is more probable that the uni-, bi-, and tri-grams generated from the word have been seen in the training data as opposed to the complete word. This can remove the previously mentioned limitation of using the words as feature based approach and make the system more scalable and computationally less expensive.

4 Experiments and Results

4.1 Experimental Settings

We experimented in four different settings, named 'Word', 'Unigram', 'Bigram' and 'Trigram'. Each refers to a setting in which a particular feature described in Section 3.3 is used i.e. 'Word' represents the setting where complete words are used as the only feature, while 'Unigram', 'Bigram' and 'Trigram' to the setting where generated uni-, bi- and tri-gram counts are used as features respectively.

In each of the settings, we kept the same division of training and test datasets as in the ICDAR 2019 competition to make the results directly comparable.

As for the machine learning models, we used sklearn's 'DictVectorizer', 'OneHotEncoder', and 'CountVectorizer' for data transformations and 'SVC' classifier with default parameters for training and testing. The only optimization was done by setting the class_weight to 'balanced' to overcome the issue of imbalanced class distribution in the training data for some of the languages (more detail about it in the following results section).

4.2 Results and Discussion

The results of the experiments are presented in Table 4 for all four settings i.e. 'Word', 'Unigram', 'Bigram', and 'Trigram'. For comparisons, the results from the ICDAR2019 post-OCR error detection task are given in Table 3. The scores are highlighted in bold if they are better than the ICDAR2019 results for the corresponding language in each individual setting, while the best score across all four settings is underlined. If a score is both bold and underlined, it means it is the state of the art score.

	F1-Score					
	CCC	CLAM	CSIITJ	RAE1	RAE2	UVA
BG	0.77	0.68	x	x	x	x
CZ	0.70	0.41	x	x	x	x
DE	0.95	0.93	x	0.90	0.89	x
EN	0.67	0.45	0.45	0.53	0.57	0.47
ES	0.69	0.56	x	0.62	0.60	x
FI	0.84	0.51	x	0.44	0.46	x
FR	0.67	0.45	0.42	0.42	0.45	x
NL	0.71	0.61	x	x	x	0.83 x
PL	0.82	0.72	x	x	x	x
SL	0.69	0.54	x	x	x	x

Table 3: Evaluation results of ICDAR2019 error detection task of the six successful systems.

	Word			Unigram						Bigram			Trigram		
							class_weight=balanced								
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
BG	0.68	0.93	**0.79**	0.83	0.80	**0.82**	0.90	0.69	**0.79**	0.89	0.84	**0.86**	0.86	0.84	**0.85**
CZ	0.93	0.12	0.21	0.95	0.57	0.70	0.83	0.73	**0.78**	0.96	0.63	**0.74**	0.96	0.51	0.67
DE	0.95	0.14	0.25	1.0	0.55	0.71	1.0	0.54	0.70	0.93	0.72	0.81	0.81	0.76	0.78
EN	0.63	0.91	**0.75**	0.89	0.71	**0.79**	0.96	0.59	**0.73**	0.89	0.78	**0.83**	0.84	0.77	**0.81**
ES	0.83	0.90	**0.86**	0.88	0.88	**0.88**	0.99	0.61	**0.75**	0.90	0.89	**0.90**	0.87	0.91	**0.89**
FI	x	x	x	0.79	0.64	0.71	0.68	0.77	0.72	0.90	0.77	0.83	0.90	0.77	0.83
FR	0.94	0.23	0.36	0.86	0.75	**0.80**	1.0	0.63	**0.77**	0.82	0.82	**0.82**	0.81	0.84	**0.83**
NL	0.80	0.95	**0.87**	0.71	1.00	**0.83**	0.98	0.45	0.62	0.71	1.00	**0.83**	0.72	0.99	**0.84**
PL	0.75	0.97	**0.85**	0.70	1.00	**0.83**	0.98	0.57	0.72	0.73	0.93	**0.84**	0.73	0.98	**0.83**
SL	0.92	0.14	0.24	1.00	0.42	0.59	0.98	0.60	**0.74**	0.97	0.44	0.60	0.93	0.32	0.48

Table 4: Evaluation results of post-error detection using the proposed methodology.

As can be seen, with 'Word', our system beats the ICDAR2019 F1 scores for five languages (BG, EN, ES, NL, PL), while ICDAR2019 scores are better for three language (CZ, DE, FI, SL). Due to the time constrains we could not complete the experiments for the remaining two languages (FI, FR).

With 'Unigram', our system beats the ICDAR2019 F1 scores for six languages (BG, EN, ES, FR, NL, PL) with the default class distribution (more details about class distribution to follow). With the balanced class distribution , we were able to beat scores of two more languages (CZ, SL) making it eight in total. For the remaining two languages (DE, FI), ICDAR2019 results are better. Also not that we get noticeable improvements in F1 scores with 'Unigram' as compared to 'Words' for all of the languages except NL and PL.

With 'Bigram', our system beats the ICDAR2019 results for seven languages (BG, CZ, EN, ES, FR, NL, PL) with default class distribution. Again, due to time constraints, we were unable to complete the experiments with the balanced class weight in the 'Bigram' setting, but similarly to the other results in the 'Unigram' setting, we should expect an improvement once the experiments have been completed. The languages for which our approach does not beat the ICDAR2019 scores are Finnish and Danish. There is a marginal difference for Finnish (0.83 vs 0.84), while a notable difference for Danish (0.81 vs 0.95). Also note, we achieved further improvements compared to the 'Unigram' F1 scores.

With 'Trigram' the results start deteriorating for most of the languages, and improve for a couple of them (FR and NL) achieving state-of-the art for NL.

In summary, we were able to beat ICDAR2019 results for eight out of ten languages in different settings reaching state-of-the art F1 scores (underlined and bold) for those languages. The improvements vary from +0.2 to +2.1 for Bulgarian (0.77 to 0.86), Czech (0.70 to 0.78), English (0.67 to 0.83), Spanish (0.69 to 0.90), French (0.67 to 0.83), Dutch (0.71 to 0.87), Polish (0.82 to 0.84), and Slovenian (0.69 to 0.74).

As can be noticed in all four settings, we get comparatively low F1 scores for CZ, DE, and SL. The reason for this is the low recall resulting from imbalanced class distribution in the training data of these languages. The class distribution for the class label 0/1 is 0.19/0.81, 0.29/0.71, 0.16/0.84 and 0.20/0.80 respectively for CZ, DE, SL, and FI as also shown in the Table 2. The sklearn's SVC classifier provides an option to balance the class weight by setting its class_weight parameter to 'balanced'. With this optimization the results improved for CZ, FI, SL as shown in Table 4 in the 'Unigram' settings while declined for the rest of the languages.

5 Conclusion and Future Work

Training supervised machine learning models with large number of features is a computationally expensive task. This has been demonstrated in previous work where handcrafted features were considered at the expense of high computational costs. In this study we have taken a different approach and have proposed to use n-gram counts as the only feature to train SVM models. N-gram counts have previously been used for post-OCR detection, but not in the sense that we have proposed in this study. Instead of computing the n-gram counts over the entire training data we have proposed to compute them within a given token and use them as the only feature to train and test our models.

To find the best 'n' for the n-grams, we experimented with uni-grams, bi-grams, and tri-grams. We tested the approach on the ICDAR 2019 dataset. As the experiment results show, with uni-grams our model outperforms the best system reported in the ICDAR2019 competition for six out of the 10 European languages included in the competition with default class distribution, and two more languages with balanced class distribution. With bi-grams, our model outperforms the previous system for 7 languages with comparable results for the remaining three languages. With tri-grams the results start deteriorating for most of the languages meaning that bi-grams are the best choice. Overall, with this approach we were able to beat 8 out of the 10 languages achieving state-of-the-art results for those languages.

The proposed approach is interesting because it eliminates the need for feature engineering; a task which is laborious and computationally expensive. The results show simple n-gram counts, which are fairly easy to compute, are enough for the task at hand. The approach is also gainful because it does not require large amounts of data. Given the relatively small datasets we experimented with we were able to show our method is performing better for the majority of languages compared to deep learning systems such as the ones explored by the CCC and UVA teams.

As previously said, we have used an SVM classifier with default parameters. In the future, we plan to apply parameter optimization, e.g. scaling, class distribution, grid-search, etc. to try and improve the results further. For example to detect real word errors when the word with an error is still a good word but inappropriate in the context. Another possible way to improve the results is to use the back-off approach in the n-gram setting. Taking a back-off approach we will use a bi-gram if a tri-gram is not in the vocabulary in a tri-gram setting, and likewise a uni-gram if a bi-gram is not in the vocabulary.

With these reported state-of-the-art results on post-OCR error detection, it will be interesting to experiment how these results will contribute to the improvement of the overall task of post-OCR error correction. We leave this to be another possible direction to explore further.

Acknowledgments

The work presented here was funded by (1) the *Dictionary/Grammar Reading Machine: Computational Tools for Accessing the World's Linguistic Heritage* (DReaM) Project awarded 2018–2020 by the Joint Programming Initiative in Cultural Heritage and Global Change, Digital Heritage and Riksantikvarieämbetet, Sweden; (2) the Swedish Research Council as a part of the project *South Asia as a linguistic area? Exploring big-data methods in areal and genetic linguistics* (2015–2019, dnr 421-2014-969); (3) *From Dust to Dawn: Multilingual Grammar Extraction from Grammars* project funded by Stiftelsen Marcus och Amalia Wallenbergs Minnesfond 2007.0105, Uppsala University; (4) the Swedish Research Council as part of the project *Evaluation and refinement of an enhanced OCR-process for mass digitisation* (2019–2020, dnr IN18-0940:1 and 421-2014-969). It is also supported by Sprkbanken Text and Swe-Clarin, a Swedish consortium in Common Language Resources and Technology Infrastructure (CLARIN) Swedish CLARIN (dnr 821-2013-2003).

References

Haithem Afli, Loïc Barrault, and Holger Schwenk. 2016. OCR error correction using statistical machine translation. *International Journal of Computational Linguistics and Applications* 7(1):175–191.

Chantal Amrhein and Simon Clematide. 2018. Supervised OCR error detection and correction using statistical and neural machine translation methods. *Journal for Language Technology and Computational Linguistics (JLCL)* 33(1):49–76.

Sandhya Arora, Debotosh Bhattacharjee, Mita Nasipuri, L. Malik, M. Kundu, and D. K. Basu. 2010. Performance comparison of SVM and ANN for handwritten devnagari character recognition. *arXiv preprint* abs/1006.5902. https://arxiv.org/abs/1006.5902.

Guillaume Chiron, A. Doucet, Mickaël Coustaty, and Jean-Philippe Moreux. 2017. ICDAR2017 competition on post-OCR text correction. *14th IAPR International Conference on Document Analysis and Recognition (ICDAR)* 01:1423–1428.

Dana Dannélls and Simon Persson. 2020. Supervised OCR post-correction of historical Swedish texts: What role does the OCR system play? In Sanita Reinsone, Inguna Skadina, Anda Baklane, and Janis Daugavietis, editors, *Proceedings of the Digital Humanities in the Nordic Countries 5th Conference, Riga, Latvia, October 21-23, 2020*. CEUR-WS.org, volume 2612 of *CEUR Workshop Proceedings*, pages 24–37.

Dana Dannélls and Shafqat Mumtaz Virk. 2020. OCR error detection on historical text using unifeature and multi-feature based machine learning models. In *Proceedings of the SLTC 2020*. University of Gothenburg, Gothenburg. https://gubox.app.box.com/v/SLTC-2020-paper-14.

Senka Drobac, Pekka Kauppinen, and Krister Lindén. 2017. OCR and post-correction of historical Finnish texts. In *Proceedings of NoDaLiDa 2017*. Linkping University Electronic Press.

John Evershed and Kent Fitch. 2014. Correcting noisy OCR: Context beats confusion. In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*. Association for Computing Machinery, New York, NY, USA, DATeCH, pages 45–51.

Norhidayu Abdul Hamid and Nilam Nur Amir Sjarif. 2017. Handwritten recognition using SVM, KNN and Neural Network. *ArXiv pre-print* abs/1702.00723. https://arxiv.org/abs/1702.00723.

David Hull and Gregory Grefenstette. 1996. Querying across languages: A dictionary-based approach to multilingual information retrieval. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, New York, NY, USA, pages 49–57.

Gitansh Khirbat. 2017. OCR post-processing text correction using simulated annealing (OPTeCA). In *Proceedings of the Australasian Language Technology Association Workshop*. Association for Computational Linguistics, Brisbane, Australia, pages 119–123.

Ido Kissos and Nachum Dershowitz. 2016. OCR error correction using character correction and feature-based word classification. In *Document Analysis Systems (DAS) 12th IAPR Workshop*. IEEE, Santorini, Greece, pages 198–203.

Okan Kolak and Philip Resnik. 2005. OCR post-processing for low density languages. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*. Association for Computational Linguistics, Vancouver, B.C., Canada, pages 867–874.

Jie Mei, Aminul Islam, Yajing Wu, Abidalrahman Mohd, and Evangelos E Milios. 2016. Statistical learning for OCR text correction. *arXiv preprint* abs/1611.06950. https://arxiv.org/abs/1611.06950.

Thi-Tuyet-Hai Nguyen, Mickal Coustaty, Doucet Antoine, and Nhu-Van Nguyen. 2018. Adaptive edit-distance and regression approach for post-OCR text correction. In *20th International Conference on Asia-Pacific Digital Libraries, ICADL*. Springer International Publishing, Hamilton, New Zealand.

Thi Tuyet Hai Nguyen, Adam Jatowt, Mickael Coustaty, and Antoine Doucet. 2021. Survey of post-ocr processing approaches. *ACM Computing Surveys* 54(6). https://doi.org/10.1145/3453476.

Thi-Tuyet-Hai Nguyen, Adam Jatowt, Mickaël Coustaty, Nhu-Van Nguyen, and Antoine Doucet. 2019. Deep statistical analysis of OCR errors for effective Post-OCR processing. In *Proceedings of the 18th Joint Conference on Digital Libraries*. IEEE Press, Champaign, Illinois, pages 29–38.

Thi-Tuyet-Hai Nguyen, Adam Jatowt, Mickaël Coustaty, Nhu-Van Nguyen, and Antoine Doucet. 2019. Post-OCR error detection by generating plausible candidates. In *International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, Sydney, Australia, pages 876–881.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research* 12:2825–2830.

Christophe Rigaud, Antoine Doucet, Mickaël Coustaty, and Jean-Philippe Moreux. 2019. ICDAR 2019 competition on post-OCR text correction. In *Proceedings of the 15th International Conference on Document Analysis and Recognition*. Sydney, Australia, pages 1588–1593. https://hal.archives-ouvertes.fr/hal-02304334.

Sarah Schulz and Jonas Kuhn. 2017. Multi-modular domain-tailored OCR post-correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Copenhagen, Denmark, pages 2716–2726.

D Sindhu and B Sagar. 2017. Dictionary based machine translation from Kannada to Telugu. *IOP Conference Series: Materials Science and Engineering* 225(1):012182.