Annotation of Struck-out Text in Handwritten Documents

Hiqmat Nisa
School of Computing Technologies, RMIT University
Melbourne, Australia
hiqmat.nisa@student.rmit.edu.au

Vic Ciesielski
School of Computing Technologies, RMIT University
Melbourne, Australia
vic.ciesielski@rmit.edu.au

James A. Thom
School of Computing Technologies, RMIT University
Melbourne, Australia
james.thom@rmit.edu.au

Ruwan Tennakoon
School of Computing Technologies, RMIT University
Melbourne, Australia
ruwan.tennakoon@rmit.edu.au

ABSTRACT

Annotating handwritten documents for training deep learning models is a major issue in handwritten text recognition. It requires manual effort to annotate each word in a document to specify the ground truth. Often documents contain struck-out text which needs to be ignored by the recognition process. In preparing training data, struck-out text needs to be represented in a way that can help deep learning models to learn to deal appropriately with the strike-outs. The question is how to do this. In this paper, we have investigated two approaches for struck-out text annotation: (1) provide no annotation, thus reducing the annotation burden, and (2) mark the struck-out text with a special symbol, we have used the symbol #. We have trained two models on a synthetically generated dataset using a convolutional neural network and LSTM. We obtained 8.8% and 9.0% character error rates for models one and two respectively. There was no statistically significant difference in the performance of the two models. This indicates that a model trained with minimal annotations can perform as well as a model trained with extra annotations for struck-out text.

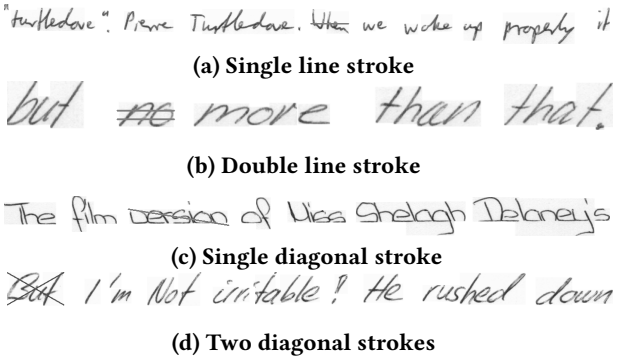
KEYWORDS

Handwritten Text Recognition, Struck-out text, Deep Learning

ACM Reference Format:
Hiqmat Nisa, Vic Ciesielski, James A. Thom, and Ruwan Tennakoon. 2021. Annotation of Struck-out Text in Handwritten Documents . In *Australasian Document Computing Symposium (ADCS '21), December 9, 2021, Virtual Event, Australia.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3503516.3503532

1 INTRODUCTION

Current Handwritten Text Recognition (HTR) systems can transcribe constrained handwritten text fairly well. Although HTR is still considered to be an unsolved problem, some recognizers are producing reasonably good results [3, 4, 14]. These recognizers expect that the handwritten document has been written in a specified, constrained format. A handwritten text collection of such format, the IAM dataset, has been produced in a controlled situation [10]. This collection is widely used in HTR systems to train a recognizer and to test whether it works on unseen data.



(a) Single line stroke

(b) Double line stroke

(c) Single diagonal stroke

(d) Two diagonal strokes

Figure 1: Different types of strike-out strokes from the Modified-IAM training set

In general, however, we cannot make any assumptions about the input handwritten document. Therefore, the HTR task is still a hard problem on unrestricted handwritten text. For example, a student's handwritten exam paper or a page of class notes, fed to a handwritten text recognizer will probably give unsatisfactory output. One of the main issues could be that recognizers usually consider dark pixels as text. However, dark pixels could come from background prints, smudges or physical damage during the scanning process. Practically, such elements are ignored while reading as they hold no information for the reader. Such elements have to be represented in a way that recognizers can differentiate between regular text and such elements.

A common element is struck-out (or crossed out) text. Struck-out text is expected to be ignored while reading, so we have to remove the struck-out item before recognizing the regular text or give them special representation during training and remove them later. Struck-out item detection and removal is a hard problem itself as it is required to segment the document first as reported by [1, 3].

Struck-out items come in different lengths and shapes, for instance, long words will be crossed out with long strokes while short words like articles and prepositions will be crossed out with short strokes. Regarding shape, people have different styles for crossing out the unnecessary text in the document. Examples include, diagonally crossed lines, a single line stroke in the middle of a word

ADCS '21, December 9, 2021, Melbourne, Australia

ACM ISBN 978-1-4503-9599-1/21/12…$15.00
https://doi.org/10.1145/3503516.3503532

Table 1: Models and their training and test data

Model	*Training data*	*Test set1*	*Test set2*	*Label*
ModelWith#	Extended_IAM_with#	IAM	Extended_IAM_with#	Struck-out text annotated as #
ModelWithout#	Extended_IAM_without#	IAM	Extended_IAM_without#	No annotation for struck-out text

and circles around a work. Sometimes we are are not able to name the stroke because the shape is so irregular. Some examples of strike-outs can be seen in Figure 1. The large variety in shapes and lengths lead us to question that whether recognizers will be able to associate the same symbolic representation to all the different representations of struck-out text. Nisa et al. [11] reported that deep learning models which are trained only on clean data, will produce inaccurate output on text containing cross outs. However, if data containing struck-out texts is provided during training, the accuracy improves. To train models on such data needs a manual annotation which is a huge task itself.

The primary goal of this work is to investigate the differences between using annotated and non annotated training data in terms of model development and test recognition accuracy.

In this work, we generated a synthetic database containing crossed-out text from a well known line version of the English language database IAM [10]. We generated some common types of strike-out strokes, depicted in Figure 1: A horizontal straight line through the middle of the word, two parallel strokes crossing the text, a single diagonal stroke and two diagonal strokes crossing each other in the middle of the word. We generated two versions of the annotated data: i.e. with # and without # for strike-outs. We trained two models on the same dataset but with the two different annotations. The architecture we used is based on a widely used handwritten text recognition system archtecture, that is Convolutional Recurrent Neural Networks (CRNN). A CRNN consists of a convolutional NN for feature extraction and an LSTM for sequence learning.

The paper organized as follows: Section 2 presents related work on struck-out text identification. The experimental methodology is explained in Section 3 and results analysis and conclusion are presented in Sections 4 and 5, respectively.

2 RELATED WORK

A very few researchers have worked on handling struck-out text in handwritten documents. For instance, Chaudhuri and Adak [3], presented a method to identify and clean struck-out words. In this method, the document is segmented based on Connected Components (CCs) of the text and then these components are tested by a classifier to identify as struck-out or not. After detection as struck-out, a struck-out stroke is identified by a graph method and removed. This method requires a whole document to be segmented first, based on CCs. This in itself is an error-prone process. Moreover, for the classification task, all the CCs have to be annotated as struck-out or not struck-out text. In a different approach, the influence of the struck-out text on the writer identification task is determined [1, 2]. Both methods are required to segment the document based on connected components. The work in [1] is only limited to straight lines and [2] only consider the longest strokes over the document.

All of the above work considers struck-out text identification as a binary classification problem that requires each CC to be labelled as struck-out or not struck-out. There are two main issues in this approach. First, the struck-out text does not always contain a whole word. Sometimes a part of the word is crossed and does not always start from the right side of the word. Segmentation of such cursive and joined words could be hard and error-prone. Secondly, all the CCs need to be annotated as struck-out text and not a struck-out text which requires manual effort. In this paper, we investigate an approach that can reduce the burden of annotation. Also, the text recognition is at the line-level which does not require word based segmentation.

3 METHODOLOGY

3.1 IAM database

Previous English handwritten text recognition systems have used the IAM database to train and test different deep learning models. In this database, annotation and segmentation are provided at the word, line, and paragraph level. There are 9,862 text lines which are further split into 6161 lines for training, 900 lines for validation1, 940 lines for validation2, and 1861 lines for testing. The database consists of 82227 words which includes 10841 unique occurrences of words and 79 different symbols. These 79 symbols consist of lowercase and capital letters, punctuation marks, digits, and some special symbol (#). We have used this dataset to train a baseline system.

There are total 71 lines in the IAM database which already contain struck-out text. Most of them are annotated with a symbol # but some of them are left unannotated. We have observed 20 unannotated crossed-outs that appear in the validation1 and test set. Therefore, the total number of struck-out text samples is 12 in the validation1 and 15 in the test set. Examples from IAM train set can be seen in Figure 3. The number of lines having no struck-out will be 6117, 888 and 1825 in training, validation1 and test set respectively. All these is shown in the first row of Table 2. This small quantity of struck-out samples in the IAM database is not enough for training the model.

3.2 Synthetic Database

As indicated earlier, the IAM dataset was constructed by having volunteers reproduce a fixed typewritten text in their own handwriting. As a result, the handwritten samples are free from editing operations, for example, crossing out inappropriate text. However, when we think and write simultaneously we do make mistakes. In contrast, handwritten samples from the IAM database are clean and organized.

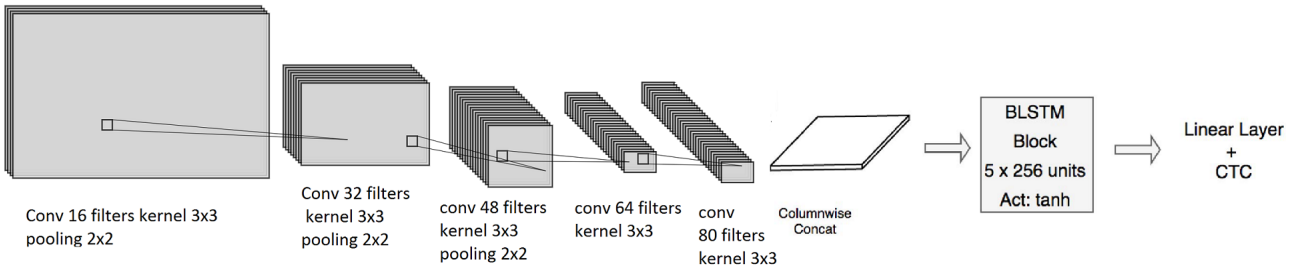


Figure 2: Network Architecture used in this paper based on [12].

A database having struck-out text is required for our work. Due to the unavailability of such a database, a data collection process needs to be carried out. There are two different ways to do this. First, real handwritten samples can be collected from humans. Second, synthetic data can be generated. The latter can be automated to quickly generate the text transcriptions. For this paper, a synthetic struck-out text was generated from the publicly available English handwritten IAM [10] database.

For better performance, it is necessary to provide a sufficient number of struck-out words to train the model so that the model can learn different patterns of struck-out words. Therefore, we have generated different types of strokes on random words from the IAM database. For this purpose, a bounding box around the target random word was constructed in a line using the line segmentation method defined by Manmatha and Srimal [9]. The bounding box comprises the whole area of the word. We used four common types of strokes. From these four types, one sample is selected randomly and superimposed on the word within the chosen bounding box. From the IAM database, 30% of the training and 25% of the validation lines were used to produce crossed-outs. Regarding distribution of crossed-outs, our training set consists of 2000 text lines having synthetically generated struck-out words, and 44 lines having struck-out text are from the IAM database. The validation set contains 295 lines having struck-out text in which 12 lines from the IAM database. Finally, the test set contains 495 lines having struck-out text in which 15 are from the IAM database. All these details can be seen in Table 2. All these text lines contains exactly one crossed-out word.

We consider the following parameters for cross-out generation.

i. Strike-out stroke style: In this paper commonly used strokes have been considered, namely, single line stroke Figure 1(a), double-line stroke Figure 1(b), single diagonal stroke Figure 1(c) and two diagonal strokes crossing each other Figure 1(d). Straight lines are used to draw all these strokes.

ii. Strike-out stroke color: For more realistic struck-out text, it is essential to match the color of the stroke with the written text in the image. Therefore, a histogram of the grayscale image was calculated to find the correct color value for strokes. The value having the highest frequency was chosen for the stroke.

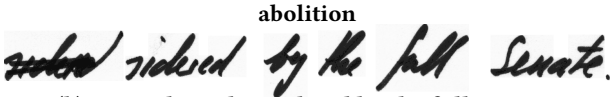
iii. Strike-out stroke width: Cross-outs in handwritten text are usually done by the same instrument that were used for the writing. The width of the writing strokes is determined by the tip of the pen/pencil. To generate more genuine strokes, stroke width should be similar to the written text width. Therefore, the average width of writing strokes was selected. For that purpose, all strokes in terms of connected components were measured. Then the Euclidean distance transform [5] was applied to each connected component. For each pixel on the stroke, the transform calculates the distance from that pixel to the nearest boundary pixel. By doing this, we can find all the points that share the maximum distance from the stroke to the boundary. The average of all these distances defines the width of the stroke.

For the struck-out words, we have two annotation choices: markup the struck-out word with a special symbol in the transcription or ignore the struck-out words in the transcription.

3.2.1 Extended_IAM_with#. In this version, the annotation of these struck-out words is represented as the symbol #. Example of such case is shown in Figure 4(a). This will help to determine the performance of the handwritten text recognition system in terms of detecting struck-out text and enable the calculation of false positive and false negative rates for struck-out text detection.

3.2.2 Extended_IAM_without#. In this version, the struck-out text is ignored in the annotation. No label is provided in the annotation to represent the struck-out words. A sample of such case is shown in Figure 4(b). In this version, we also removed the # labels of struck-out words that were already available in the IAM database. The purpose of this version is to investigate the ability of the HTR system to ignore the struck-out text. In this way, we can assess whether the HTR system can give attention only to regular text and ignore the struck-out text even when it is crossed out with one stroke only.

(a) Ground Truth: sentiment would # still favour the abolition



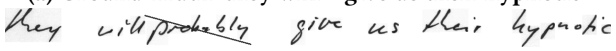
(b) Ground Truth: # sidered by the full Senate.

Figure 3: Examples of struck-out text from IAM database

Table 2: Data Sets

Database	Lines						Words		Characters	
	Training		Validation		Test set		Training	Test set	Training	Test set
	Non struck-out	*Struck-out*	*Non struck-out*	*Struck-out*	*Non struck-out*	*Struck-out*				
IAM	6117	44	888	12	1825	15	53807	17403	287515	83340
Extended_IAM_with#	6117	2044	605	295	1345	495	71460	17403	376535	82900
Extended_IAM_without#	6117	2044	605	295	1345	495	69311	16921	372338	81835

(a) Ground Truth: they will # give us their hypnotic



(b) Ground Truth: they will give us their hypnotic

Figure 4: Two version of annotation of same text line. Extended_IAM_with# (a) and Extended_IAM_without# (b)

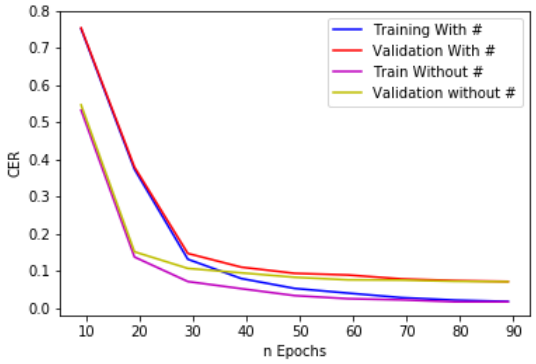


Figure 5: CER of two different Models

3.3 Model

Earlier HTR systems were based on Multi-Dimensional LSTM (MDLSTM) architecture which is a combination of Multi-Dimensional LSTM (MDLSTM) layers and convolutional layers [13]. The use of MDLSTM layers is expensive in terms of memory and time [7]. To overcome this issue, researchers replaced the MDLSTM with Bidirectional LSTM (BLSTM) which reads the input forward and backward only.

In this work, we used a Bidirectional LSTM (BLSTM) based model which was originally proposed in [11], as it has achieved state-of-the-art results in HTR. An overview of the network architecture is given in Figure 2. The network contains five convolutional layers with 16, 32, 48, 64 and 80 filters respectively. Each kernel is 3x3 pixels with a stride of 1x1 is used in each convolutional layer. Each convolution layer is followed by a Leaky Rectifier Linear Units (LeakyReLU) activation function. a max-pooling layer with a kernel size of 2x2 and stride of 1 is used after the third convolutional layers to reduce the size of the input. After the last convolutional layer, five layers of BLSTM each containing 256 hidden units were used. As LSTM is bidirectional, we performed a depth-wise concatenation at the end of each BLSTM to adapt the next layer input format. To reduce overfitting, dropout regularization was applied with the probability of 0.2 at the output of each convolutional layer except the first and with the probability of 0.5 at the output of each LSTM layer.

Finally, a fully connected layer is used for the predictions. This prediction layer contains the same number of nodes as the number of characters in the IAM database. We have used 80 nodes, 79 are characters and one additional node is needed for the blank symbol of CTC [6].

3.4 Preprocessing and data augmentation

Pre-processing of images includes binarization and resizing. We made all the images to the same height while maintaining their aspect ratio and padded them at the end to made them equal in width. To artificially increase the amount of training data, we augmented the existing images by applying minor changes. Different types of distortion like dilation, erosion, shear, translation, and rotation were combined to simulate commonly occurring variations in handwritten text line images. These distortions were applied to the original line image randomly with an independent likelihood of 50%. No language model was embedded during training.

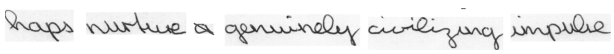
3.5 Implementation

The implementation of architecture used in this paper was done in the Tensorflow framework. We used the ADAM optimizer with learning rate 0.001. The gradients of CTC loss were used to update the parameters on a batch size of 20 text lines. For the evaluation, Character Error Rate (CER) and Word Error Rate (WER) were used to evaluate the performance of the recognizer. CER and WER are calculated by implementing the Levenstein edit distance [8] which counts the number of edit operations needed to convert one string into another string. In the calculation of WER, a word is defined as anything that is separated by spaces in the transcription of the handwritten document. For instance, it includes full stops, commas and semi-colon appear at the end of the word.

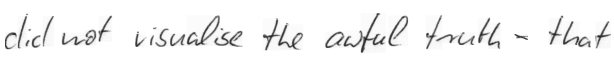
3.6 Experiment with Extended_IAM_with#

As mentioned in the introduction, that struck-out texts come in different styles and shapes. So, by providing the same symbolic annotation for all types of strike-outs, we expect that the CRNN model will be able to generalize the features of these strike-outs and detect them as symbol #. For this purpose, we have trained our model, ModelWith#, on Extended_IAM_with#. Details of this model are presented in the first row of Table 1.

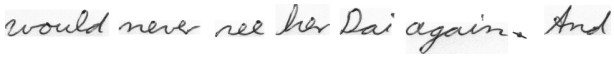
During training, we regularly checked the graphs of loss and accuracy. We stopped training after 90 epochs because the loss on the validation data set started to increase indicating over-fitting. On the training and validation sets, we achieved 2.0% and 7.0% CER, respectively. The training and validation CERs can be seen in Figure 5. After the completion of training, the trained model was tested on the Extended_IAM_with# testset for struck-out text detection. This test set contains 495 strike-outs of which 440 were identified correctly. The remaining 55 were not detected as strike-outs (see the confusion matrix in Table 4. Almost all of these false negatives have punctuation marks or a single letter crossed-out with a single stroke. Some of the examples can be seen in Figure 6. We have also seen exactly 12 false positives where the text is not struck-out but detected as struck-out text. Most of the false positives contain short words consisting of letters possessing lines in them, such as letter 't' and 'H'. From this experiment, we have observed that in our case the CRNN model was unable to associate the same symbolic annotation for all styles and shapes crossed-outs. Short words and a single line stroke were hard to detect for the model.



Ground Truth: haps nurture # genuinely civilizing impulse
Predicted: haps nurtue a genuinely civilizing implle



Ground Truth: did not visualise the awful truth # that
Predicted: did not visualise the auful truth- that



Ground Truth: would never see her Dai again # And
Predicted: would never nee her Pai again- And

Ground Truth: black eyes showing # emotion. With a
Predicted: black eyes shosing any enstion. With a

Figure 6: Some examples of false Negatives

3.7 Experiment with Extended_IAM_without#

The purpose of this experiment is to investigate that whether the CRNN model can ignore the struck-out words and only focus on regular text. For this purpose, no annotation for struck-out text is provided. We simply ignore the struck-out text during transcription. The same architecture and parameters were used as for Extended_IAM_with#. Training progress comparison of two models can be seen in Figure 5. We achieved 2.0% and 7.0% CER on training and validation sets respectively.

3.8 Discussion

The first row of the Table 3 shows the CER (7.0%) and WER (22.0%) of a reference model that was trained the IAM database without any synthetical dataset. We have also represented a CER and WER of the two different models trained on the same dataset with different annotations for struck-out text. For a fair comparison, we have selected both models based on the same number of epochs. These three models were tested on three different test sets. The IAM test set, and Extended_IAM_without# and Extended_IAM_with# test sets. The first test set is the IAM dataset, the purpose of testing models on this test set is to look for false positives. As this set does not have any synthetic struck-out text, assessing this set will give us a fair idea about the performance of the models in terms of false struck-out text detection. Figure 7 shows some of these examples where the same images are tested on both models. ModelWith# produced # when the text looks like having a stroke/s. In contrast, ModelWithout# gives an alternative letter.

Our second test set, Extended_IAM_without# has cross outs without annotation. The IAM model was not trained on synthetical strikethrough data, so we were expecting the increase in CERs when testing on Extended_IAM_without#. We also observed a slight increase in CERs in both models' performance as compared to IAM test set which shows that introducing struck-outs can degrade the performance of the models whether the struck-out text is annotated or not. However, a comparison of these models based on Extended_IAM_without# gives us a fair idea whether the struck-out texts should be marked for training data or not. As we know that Modelwith# is trained on struck-out as #, testing this model on Extended_IAM_without# will produce # in the specific position where the text is crossed-out in the original image. All the # removed from the transcription and then we calculate the CER and WER. We achieved 8.8% CER and 9.0% on ModelWith# and ModelWithou# respectively. From the result, we can say that a model can perform slightly better if the struck-out text is annotated during training. Nevertheless, the difference between the accuracy is too small, so we have to perform a statistical test to check whether the difference is statistically significant or not.

Our third test set, Extended_IAM_with# has cross outs with annotation. As this test set is having representation for crossed-outs so we can only check the performance of the ModelWith#. Overall, we can see from the table 3 that ModelWith# outperform when it was tested on the Extended_IAM_with# test set as compare to ModelWithout# when it was test on Extended_IAM_without#.

4 RESULTS ANALYSIS

To check whether the difference in accuracy between Modelwith# and ModelWithout# is statistically significant, we conducted a Wilcoxon signed rank test. Statistical tests require repeated observations, but we only have one test set. As we know, handwritten text lines in the IAM test set were written by 128 different authors.

Table 3: CER and WER for the two models

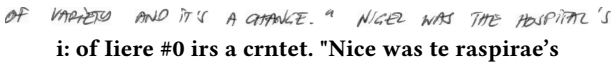
		IAM Test set 1		Extended_IAM_without# Test set 2		Extended_IAM_with# Test set 2	
	Training data	***CER***	***WER***	***CER***	***WER***	***CER***	***WER***
IAM	IAM Training set	7.0	22.0	11.0	25.0	11.0	25.0
ModelWith#	Extended_IAM_with#	8.3	22.1	8.8	23.0	8.6	22.2
ModelWithout#	Extended_IAM_without#	8.5	22.5	9.0	23.2	-	-

Table 4: Confusion matrix on Extended_IAM_with# test set for ModelWith#

	Actual	
	Struck-out	**Not struck-out**
Struck-out	440	12
Not Struck-out	55	16893

i: openness and #hness, though perhaps not the some
ii: openness and Rahness, though perhaps not the some

i: Puurrce masim opercen. "Jon Nowir # # asour WWe, I
ii: "Puwice mesiem opprcen. 'Son howr' asour WWe, I



i: of Iiere #0 irs a crntet. "Nice was te raspirae's
ii: of mprers nd its a aube. "NOR was The rapitare's

Figure 7: Some examples from the IAM test set. i: Result of ModelWith# ii: Result of ModelWithout#

We divided this test set into 128 sets according to 128 authors. The graph in Figure 8 a comparison of the two model CERs for each of the 128 authors. From this graph we can see that the CERs are not fully overlapped which shows that for the same handwritten text these models did not produce the same results.

Our data is not normally distributed, so we performed a non-parametric test, the Wilcoxon signed rank test which does not assume normal distributions. We set our null hypothesis that there is no statistically significant difference between the accuracies of the two models. We used .05 p-value for this test. The p value was 0.260 which is considerably greater than 0.05. Based on a significance level of p= .05, there is a no statistically significant difference between the accuracy of ModelWith# and ModelWithout#.

5 CONCLUSION

In this paper, we have presented an approach to deal the annotation of struck-out text in handwritten manuscripts. In this approach, we have trained two models on the same dataset with two different annotations for struck-out text. We used a dataset consisting of handwritten text lines having synthetically generated crossed-out text from the widely used IAM database. The first model was trained

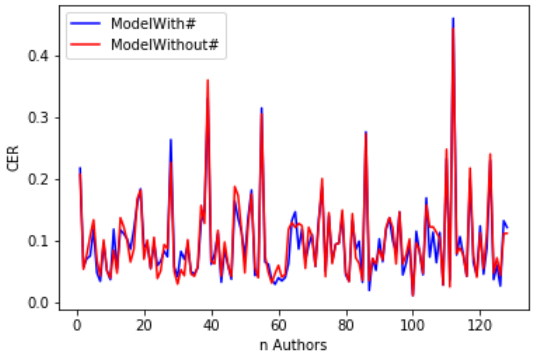


Figure 8: CERs of 128 Authors calculated from two model

on a dataset in which struck-out was annotated as symbol # to answer the question of whether recognizers will be able to associate the same symbolic representation to all the different sizes and shapes of struck-out text. From the results, we have concluded that there is no difference if the strikethrough text is annotated or not. It is found that this model was unable to detect crossed-out text where a single line stroke is used to cross out the punctuation marks or a very short word. The second model was trained without providing any annotation for struck-out text during training to determine whether the recognizer is able to learn only regular text and not learn the struck-out text. We found that CRNN model can ignore the struck-out text during training if no annotation is provided.

Finally, we have also did the comparison of these two models on a test set without annotation for struck-out. From this comparison, we conclude that it does not make any significant difference in terms of CER if you train a model on a database having struck-out with representation in transcription or without annotation. However, train a model having annotation for struck-out text will help to assess the performance of the model for struck-out text detection. On the other hand, training a model on a database having no annotation reduces the burden of labeling a database having cross-outs.

A limitation of this work is that our strike-outs are synthetically generated. These strike-outs only represent the most common styles for a single word. However, strike-outs can be of any size. A strike out could contain more than one word, or it could be part of a word. In future work we plan to extend the work to real handwritten text.

REFERENCES

[1] Chandranath Adak, Bidyut B Chaudhuri, and Michael Blumenstein. 2017. Impact of struck-out text on writer identification. In *2017 International Joint Conference*

on Neural Networks (IJCNN). IEEE, 1465–1471.

[2] Axel Brink, Harro van der Klauw, and Lambert Schomaker. 2008. Automatic removal of crossed-out handwritten text and the effect on writer verification and identification. In *Document Recognition and Retrieval XV*, Vol. 6815. International Society for Optics and Photonics, 68150A.

[3] Bidyut B Chaudhuri and Chandranath Adak. 2017. An approach for detecting and cleaning of struck-out handwritten text. *Pattern Recognition* 61 (2017), 282–294.

[4] Arindam Chowdhury and Lovekesh Vig. 2018. An efficient end-to-end neural model for handwritten text recognition. *arXiv preprint arXiv:1807.07965* (2018).

[5] Juan C Elizondo-Leal and Gabriel Ramirez-Torres. 2010. An Exact Euclidean Distance Transform for Universal Path Planning. In *2010 IEEE Electronics, Robotics and Automotive Mechanics Conference*. IEEE, 62–67.

[6] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*. 369–376.

[7] José Carlos Aradillas Jaramillo, Juan José Murillo-Fuentes, and Pablo M Olmos. 2018. Boosting handwriting text recognition in small databases with transfer learning. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE, 429–434.

[8] Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, Vol. 10. Soviet Union, 707–710.

[9] Raghavan Manmatha and Nitin Srimal. 1999. Scale space technique for word segmentation in handwritten documents. In *International conference on scale-space theories in computer vision*. Springer, 22–33.

[10] U-V Marti and Horst Bunke. 2002. The IAM-database: an English sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition* 5, 1 (2002), 39–46.

[11] Hiqmat Nisa, James A Thom, Vic Ciesielski, and Ruwan Tennakoon. 2019. A deep learning approach to handwritten text recognition in the presence of struck-out text. In *2019 International Conference on Image and Vision Computing New Zealand (IVCNZ)*. IEEE, 1–6.

[12] Baoguang Shi, Xiang Bai, and Cong Yao. 2016. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence* 39, 11 (2016), 2298–2304.

[13] Paul Voigtlaender, Patrick Doetsch, and Hermann Ney. 2016. Handwriting recognition with large multidimensional long short-term memory recurrent neural networks. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE, 228–233.

[14] Mohamed Yousef, Khaled F Hussain, and Usama S Mohammed. 2020. Accurate, data-efficient, unconstrained text recognition with convolutional neural networks. *Pattern Recognition* 108 (2020), 107482.