On the Use of Neural Text Generation for the Task of Optical Character Recognition

Mahnaz Mohammadi[1], Sardar Jaf [2], Andrew Stephen McGough[3], Toby P. Breckon[1], Peter Matthews[1], Georgios Theodoropoulos[4], Boguslaw Obara[1]

[1]*Dept. of Computer Science*, *Durham University*, UK
[2]*School of Computer Science*, *Sunderland University*, UK
[3]*School of Computing*, *Newcastle University*, UK
[4]*Dept. of Computer Science and Eng., Southern University of Science and Technology*, China

[1] *mahnaz.mohammadi,toby.breckon,p.c.matthews, boguslaw.obara*@durham.ac.uk
[2] *sardar.jaf@sunderland.ac.uk*
[3] *stephen.mcgough@newcastle.ac.uk*
[4] *georgios@sustc.edu.cn*

***Abstract*—Optical Character Recognition (OCR), is extraction of textual data from scanned text documents to facilitate their indexing, searching, editing and to reduce storage space. Although OCR systems have improved significantly in recent years, they still suffer in situations where the OCR output does not match the text in the original document. Deep learning models have contributed positively to many problems but their full potential to many other problems are yet to be explored. In this paper we propose a post-processing approach based on the application deep learning to improve the accuracy of OCR system (minimizing the error rate). We report on the use of neural network language models to accomplish the task of correcting incorrectly predicted characters/words by OCR systems. We applied our approach to the IAM handwriting database. Our proposed approach delivers significant accuracy improvement of 20.41% in F-score, 10.86% in character level comparison using Levenshtein distance and 20.69% in document level comparison over previously reported context based OCR empirical results of IAM handwriting database.**

***Index Terms*—Neural text generation, Optical character recognition, OCR, OCR post-processing, language models, neural language model, text generation, text prediction, IAM database, handwritten character recognition**

I. Introduction

Optical Character Recognition (OCR) has facilitated the conversion of large quantities of scanned text documents, where manual conversion is not practical. Unfortunately OCR applications produce incorrect text that do not match the original text in a given document. OCR errors are due to shortcomings of OCR engines, bad physical condition (e.g. poor photocopies of the original page), poor printing quality [1] and/or existence of large amount of handwritten text in the scanned document that are not easily readable. The accuracy of OCR output depends significantly on the quality of the input image (scanned document). Pre-processing steps, such as image re-scaling; skew correction; binarization and noise removal, are some techniques to improve the quality of the input image. However, pre-processing approach may not tackle acceptable accuracy thus we propose post-processing techniques to significantly improve the accuracy of ORC. Our proposed post-processing approach utilizes deep learning model for neural text generation, which helps predicting correct output for OCR system. We have evaluated our approach on a widely used database (IAM handwriting database) and measured the performance of our solution using F-score, Levenshtein distance and document level comparison. We have consistently improved the OCR accuracy. We have achieved significant accuracy improvement of 20.41% in F-score, 10.86% in character level comparison using Levenshtein distance and 20.69% in document level comparison over previously reported context based OCR empirical results of IAM handwriting database.

The rest of this paper is organized as follows. Section II covers some of the related work. Our language models for OCR post-processing are discussed in Section III. We evaluate the performance of the language models using different length of input sequences (n-grams) for regenerating the input text and for improving the OCR results on the transcriptions of IAM handwriting database in Section IV. We conclude the paper in Section V.

II. Related Work

Different techniques have been proposed for OCR post-processing such as manual error correction, dictionary (or lexical) based error correction and context-based error correction [2]–[7]. Manual error correction of OCR output is time-consuming and error-prone. lexical based post-processing [8]–[10] verify the OCR results using lexical knowledge and generates a ranked list of possible word candidates. However, lexical post-processing techniques can become computationally inefficient if a large vocabulary is used.

Neural Networks have contributed to many challenging natural language processing problems, such as machine translation; speech recognition; syntax/semantic parsing and infor-

mation retrieval. They can be used for automatic text generation, where new sequences of characters/words with shared statistical properties as the source text could be generated. Language modelling involves predicting the next character/word in a sequence given the sequence of characters/words already present.

Recently, the use of neural networks in the development of language models has become very popular [11], [12]. Neural network approaches have often demonstrated better results than statistical or rule-based approaches both on stand alone language models and embedded models into larger applications for challenging tasks like speech recognition and machine translation. Using neural network language models to assist OCR systems in improving the recognition accuracy rate, has interested many researchers [13], [14].

In [14] a simple language model approach is used for grammatical error correction with minimal annotated data and it was shown that this approach is competitive with the latest neural and machine learning approaches that rely on large quantities of annotated data. Kissos and Dershowitz [13] have examined the use of machine learning techniques for improving OCR accuracy by using the combination of features to enhance an image for OCR and to correct misspelled OCR words. The relative independence of the features, issues from the language model, OCR model and document context, enables a reliable spelling model that can be trained for many languages and domains.

In this paper we present two neural network language models to improve the accuracy of OCR for scanned text documents containing a mixture of handwritten and machine printed texts. our models predict the probability of the next character/word in a sequence based on previous characters/words already observed in the sequence. To the best of our knowledge, this is the first work reported on applying neural network model on mixed text recognition. We apply our post-processing approach to the output of the pipeline proposed by [15] for mixed text recognition over IAM handwriting database [25] to show the effectiveness of neural network based natural language generation on the improvement of OCR accuracy.

III. Methodology

A statistical language model is a probability distribution over sequences of words. For example, for a sequence of words of length n as $w_1 w_2 ... w_n$, language models assigns a probability $P(w_1, w_2, ..., w_n)$ to the whole sequence. Neural language models use continuous representations, or embeddings of words, to make their predictions [17]. Embeddings help to alleviate the curse of dimensionality in language modelling. Training a language model on large text generally increases the number of unique words. Consequently, this leads to the exponential increase of the number of possible sequences of words, especially since the vocabulary size is often increased in large text. This exponential increase in the number of sequences causes a data sparsity problem. Thus, statistics are needed to properly estimate probabilities.

Typically, neural network language models are constructed and trained as probabilistic classifiers that learn to predict a probability distribution of work given the context of that work, as in $P(w_t|context) \forall t \in V$. The network is trained to predict a probability distribution for a word w_t over the vocabulary V, given some linguistic context $context$. The $context$ can be a fixed-size window of previous k words, so that the network predicts $P(w_t|w_{t-k}, ..., w_{t-1})$.

A word level language model predicts the next word in the sequence based on the specific words proceeding it in the sequence. It is also possible to develop language models at the character level using neural networks. In this section we present word level and character level language models for text generation and also OCR post-processing on the IAM handwriting database.

A. Word Level Language Model for OCR Post-Processing

A language model can predict the probability of the next word in the sequence, based on the words already observed in the sequence. Neural network models are preferred methods for developing statistical language models because they can use a distributed representation where different words with similar meanings have similar representation and because they can use a large context of recently observed words when making predictions. Fig. 1 shows the architecture of the word level language model we used for training over IAM handwriting database [25].

The required general steps for the development of our neural network based language model are:

- Preparing the text for a word level language model by converting it to embedding format.
- Designing and fitting a neural language model with a learned embedding and Long Short-Term Memory (LSTM) [18] hidden layers.
- Using the learned language model to predict a word given its previous context.

We prepare the training data by cleaning (removing punctuation), formalizing it to lower case to reduce the vocabulary size, tokenizing the text, building sequences of context and target words (n-grams) using NLTK [19] and encoding the sequences to integer values as the embedding layer expects input sequences to be comprised of integers [20].

Next we fit a neural language model on the prepared data. The model uses a distributed representation of words so that different words with similar meanings have a similar representation. It learns to predict the probability of the next word using the context of the last n preceding words.

To make predictions, we pre-process the testing data in same was we as we did for the training data. At each testing iteration a context sequence preceding the word which has been transcribed incorrectly by OCR will be given to the trained model as input and a word is generated by the model, which is the target word and is used for replacing the incorrect word.

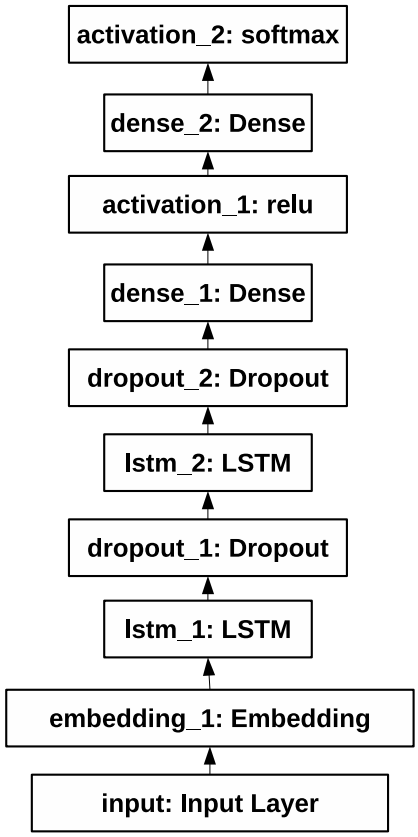

Fig. 1. The Proposed Word Level Language Model Architecture for OCR Post-Processing

1) Word Level Language Model Configuration: Our word level language model is configured as following:

- Embedding layer: An embedding layer is used to learn the representations of the words in the input sequences. We have used dimension equal to 100 for each word in the input sequence.(i.e. the embedding dimension for the model with input sequence length equal to 4 words is $4 \times 100 = 400$).
- Hidden layer: Long Short Term Memory (LSTM) recurrent neural network learns to predict words based on their context. We are using two LSTM layers. Each layer has equal number of units to the embedding dimension.
- Dropout layer: We apply dropout as a regularization technique to avoid over-fitting the model on the training data. We use a dropout layer with rate 0.5 after each LSTM layer.
- Dense layer: A dense fully connected layer with *(ReLU)* activation function connects to the LSTM hidden layer to interpret the features extracted from the sequence.
- Output layer: The output layer is a dense layer which predicts the next word as a single vector of the size of the vocabulary with a probability for each word in the vocabulary. A *(softmax)* activation function is used to ensure the outputs have the characteristics of normalized probabilities.
- Optimizer: The model is optimized with the *Adam* optimizer which need relatively low memory and works well with little tuning of hyper-parameters and we use the *categorical cross entropy loss* as the metric for computing the model's error rate.
- Epoch and batch size: Finally, we fit the model on the training data for a set of specified training epochs with different batch sizes. The ultimate hyper-parameters are decided based on the best model with lowest validation loss obtained. The validation is computed by supply a set of validation data (which is a subset of training data) to the model during training.
- The model was fit to the data with with modest number of 200 epochs and batch size of 500 patterns.

B. Character Level Language Model for OCR Post-Processing

It is possible to develop language models at character level using neural networks. The benefit of character based language models is their small vocabulary and flexibility in handling any words, punctuation, and other document structure. Building a character level language model requires converting the input sequences (characters) into a form that the model can learn from, as required in word level language model. Longer sequences offer more context for the model to learn the next output character, but the model could take longer to train. Fig.2 shows the architecture of our language model: Data preparation is the first step in developing the language model. We clean the data, normalize the case to lowercase and remove punctuation to reduce the final vocabulary size and develop a small leaning model. We build the input-output sequences of specific character lengths. The output will be the last character of each sequence. The sequences of characters are encoded as integers. Each unique character will be assigned a specific integer value and each sequence of characters will be encoded as a sequence of integers. We can create the mapping given a sorted set of unique characters in the raw input data. The mapping is a dictionary of character values to integer values. Next, we need to one hot encode each character to a vector with the size of the vocabulary. The output patterns are converted (single characters converted to integers) into a one hot encoding [20].

1) Character Level Language Model Configuration: Our character level language model is configured as following:

- Input layer: Input layer accepts the encoded input sequences.
- Hidden layers: The model has two LSTM hidden layers. Each LSTM layer has 1000 neurons (i.e. This number is finalized based on trail and error) followed by a dropout layer with rate 0.2 to avoid over-fitting the model during training.
- Output layer: The output layer is a Dense layer using the *(softmax)* activation function to output a probability prediction for each of the characters in the vocabulary between 0 and 1.

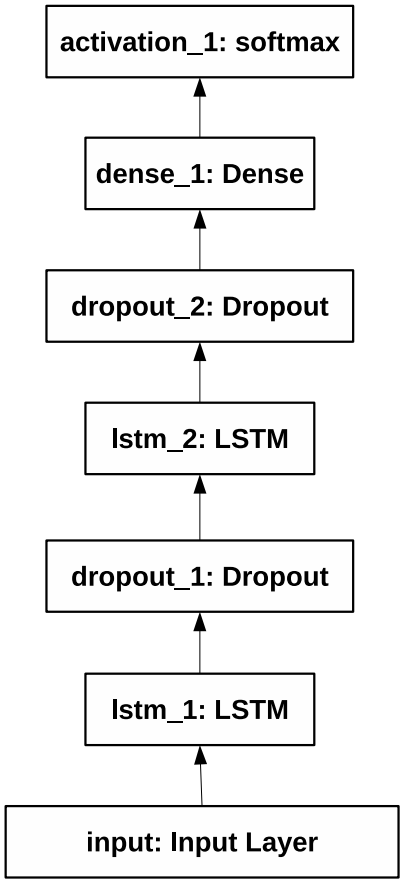

Fig. 2. The Proposed Character Level Language Model Architecture for OCR Post Processing

- Optimizer: The model is compiled specifying the *categorical cross entropy loss* and *Adam* optimizer with a relatively low learning rate of 0.001.
- We fit the model to the data with modest number of 100 epochs and a large batch size of 1000 patterns.

Finally we use the trained model to predict the next character given a sequence of characters from testing data as the input to the model.

IV. Experiments

We use several metrics to measure the performance of the proposed language models over different transcription levels: character level, word level and document level.

- *Character Level Evaluation*: The character level evaluation uses the Levenshtein distance [21], to measure the number of deletions, insertions, or substitutions required to transform the predicted document to the target document.
- *Word Level Evaluation*: Word level evaluation uses Bag-of-Word (BoW) [22] representation, which involves comparing the frequency of words occurrences in either the predicted or the target transcription disregarding the syntax structure.
- *Document Level Comparison*: Document similarity, involves generating an embedding vector [23] for the complete document of the predicted and the target transcription, and we measure the similarity rate between the generated embeddings.

A. Data Set and Analysis

The IAM handwriting database [16] is composed of 1539 different forms of English text written by 600 writers. A single page of the IAM database is split into two sections, machine-printed text in the upper section and the bottom section of the page is the repetition of the same machine-printed text written by the writers. The different writing style of the writers has been the reason for some missing parts in the handwritten section of files. The database is released with tokenized transcription for each text line of the handwriting section. Paragraph-level transcription for the machine-printed section of the page is also provided along with the database. Natural Language Processing Toolkit (NLTK) [19] is used to tokenize the machine printed section and merged both sections into a single transcription in [24].

To understand the type of the data and its complexity we have conducted the following analysis on the IAM handwriting database. Table I shows total number of characters/tokens before and after cleaning the data and also total number of unique characters/tokens after cleaning data for both train files and test files (i.e. result of applying TMIXT [15] on IAM handwriting database for text recognition). The Vocabulary size, bolded in table I, shows the number of cleaned and unique characters/tokens (words) for character level and word level language models. In the case of the word level language model the number of unique tokens for testing data is much more than the unique tokens in training data. This is due to gibberish words which have been generated during the recognition process. The model is trained on the cleaned data. Table II shows the number of handwritten and printed characters/tokens before and after cleaning data in the training files. The number of files with the same number of handwritten and printed characters/tokens and with more or less handwritten characters/tokens are also shown in this table. As presented in table II, there are quite a large number of handwritten files

TABLE I
Analysis of Train and Testing Data of IAM Handwriting Database

	Training Data	Testing Data
Total characters	1167066	1167066
Total Cleaned characters	922120	922120
Total Unique characters	82	83
Total Cleaned Unique characters	**28**	28
Total Tokens	203618	188973
Total Cleaned Tokens	200983	174542
Total Unique Tokens	19422	45910
Total Cleaned Unique Tokens	**12213**	28464

with missing characters/tokens. We found that 174 files out of 1539 (11.3%) have missing handwritten tokens in them. A total of 3 files contained missing handwritten characters. The number of missing handwritten tokens in those files were 95, 59 and 56 tokens. The maximum characters missing in the files are 11832. Some handwritten files also contained an extra number of tokens compared to the printed files. These

TABLE II
ANALYSIS OF PRINTED AND HANDWRITTEN FILES IN IAM HANDWRITING DATABASE

	Before Cleaning	After Cleaning
Printed characters	714096	466976
Handwritten characters	650997	455144
Complete Files	5	3
Files with Missing Handwritten Chars	413	1536
Files with More Handwritten Chars	1121	0
Printed Tokens	143127	101690
Handwritten Tokens	109296	99293
Complete Files	1012	1053
Files with Missing Handwritten Tokens	216	174
Files with More Handwritten Tokens	311	312

analyses describe that the type of data is quite complex for a language modelling task due to missing characters/tokens in files. The most frequent words are the stop words and there are also large numbers of words which are not frequent words (i.e. repeated less than 10 times). The total number of unique and cleaned tokens in the training data is 12213 tokens which is less than the total unique and cleaned tokens in test files (28464 tokens).

B. Experiments with Word-level Language Model

Table III shows the performance of word level language model over training data with different input sequence lengths (n-grams). The comparison has been done between the training data and the predicted data based on characters using Levenshtein distance [21], word level using F-score and document level using document similarity [23] to see how model performs in case of regenerating the training data. The obtained results, show that training model with inputs sequences of 7 words for regenerating the training data, result in higher train accuracy and consequently higher similarity rate between the training data and the data generated by the language model. For OCR post-processing using a word level language model, the input sequences from the testing data, preceding the words that are not recognized correctly via OCR will be given as inputs to the trained model. The word level language model generates a target word for each input sequence as the replacement for the word which has been recognized wrongly by the OCR. Table IV shows performance evaluation of the word level language model for OCR post-processing over the testing data (i.e. the results of applying TMIXT [15] on IAM handwritten pages).

As the results of these evaluations describe, the sequences of 5 and 6 words input lengths give higher accuracy rate for testing data. The notable difference between performance of

TABLE III
PERFORMANCE EVALUATION OVER TRAINING DATA USING PROPOSED WORD LEVEL LANGUAGE MODEL

Input \ Metric	Training Accuracy (%)	F-score (%)	Levenshtein Distance (%)	Document Similarity (%)
2 Words	62.41	20.02	29.89	30.66
3 Words	88.59	41.90	45.28	60.05
4 Words	96.09	76.86	77.75	85.07
5 Words	97.42	90.53	90.92	94.52
6 Words	97.38	92.93	93.36	96.12
7 Words	97.06	94.73	95.03	97.27
8 Words	95.96	93.62	94.00	96.66

TABLE IV
PERFORMANCE EVALUATION OVER TESTING DATA USING PROPOSED WORD LEVEL LANGUAGE MODEL FOR OCR POST-PROCESSING

Input \ Metric	F-score (%)	Levenshtein Distance (%)	Document Similarity (%)
2 Words	41.71	48.03	54.88
3 Words	56.24	58.74	71.24
4 Words	72.32	74.05	82.88
5 Words	77.79	79.36	85.03
6 Words	77.35	79.42	85.20
7 Words	76.83	78.67	84.31
8 Words	74.16	76.37	82.47

the model over training data and testing data is an evidence of the problems associated with this data such as missing words in handwritten files, gibberish words created via OCR and its complexity for language models.

C. Experiments with Character-level Language Model

Tables V and VI show performance evaluation of the character level language model over training and testing data of IAM handwriting database with different input sequence lengths. The input to the character based language model is the n-grams of characters with different lengths. As the results depict, longer input sequences result in improved performance of the model for regenerating the training data and OCR post-processing. In the case of IAM handwriting database, input sequences of length 50 characters result in higher similarity between the training data and the generated data by the model.

TABLE V
PERFORMANCE EVALUATION OVER TRAINING DATA USING PROPOSED CHARACTER LEVEL LANGUAGE MODEL

Input \ Metric	Training Accuracy (%)	F-score (%)	Levenshtein Distance (%)	Document Similarity (%)
15 Chars	98.36	70.59	75.32	84.47
20 Chars	98.76	88.14	89.87	94.22
25 Chars	98.73	90.00	91.54	95.19
30 Chars	98.38	91.01	92.43	95.81
40 Chars	98.23	86.19	88.66	93.60
50 Chars	98.91	91.79	93.18	96.52

TABLE VI
PERFORMANCE EVALUATION OVER TESTING DATA USING PROPOSED CHARACTER LEVEL LANGUAGE MODEL FOR OCR POST-PROCESSING

Input \ Metric	F-score (%)	Levenshtein Similarity (%)	Document Similarity (%)
15 Chars	72.34	77.26	89.18
20 Chars	82.80	90.00	93.66
25 Chars	84.01	85.99	94.25
30 Chars	85.62	87.18	95.17
40 Chars	82.52	84.92	94.03
50 Chars	86.87	88.03	95.75

D. Distribution of Documents Across Different Accuracy Ranges

Tables VII and VIII show the distribution of the documents across different accuracy ranges using the Levenshtein distance and document similarity metrics for comparing the training data and generated data via the word level language model. These distributions have been reported for evaluating the model with both training and testing data using different input sequence lengths. Decreasing or increasing the sequences length beyond 6 or 7 words result in degrading the performance of the language model in predicting the correct word.

TABLE VII
NUMBER OF DOCUMENTS IN DIFFERENT ACCURACY RANGES BASED ON CHARACTER SIMILARITY USING LEVENSHTEIN DISTANCE METRIC FOR WORD LEVEL LANGUAGE MODEL. WRD=WORDS

Range \ Input	2 Wrd	3 Wrd	4 Wrd	5 Wrd	6 Wrd	7 Wrd	8 Wrd
90%-100%	5	213	751	979	998	1008	953
80%-90%	6	76	77	49	48	41	47
70%-80%	70	176	109	59	49	50	67
60%-70%	415	323	149	85	62	51	21
50%-60%	213	158	72	35	36	14	10
40%-50%	279	186	64	18	13	13	39
30%-40%	264	132	31	14	25	31	368
20%-30%	287	275	285	300	308	341	1

TABLE VIII
NUMBER OF DOCUMENTS IN DIFFERENT ACCURACY RANGES BASED ON DOCUMENT SIMILARITY METRIC FOR WORD LEVEL LANGUAGE MODEL. WRD=WORDS

Range \ Input	2 Wrd	3 Wrd	4 Wrd	5 Wrd	6 Wrd	7 Wrd	8 Wrd
90%-100%	31	64	247	326	1071	1074	997
80%-90%	240	314	642	742	92	72	97
70%-80%	225	418	190	100	26	9	23
60%-70%	191	190	90	38	16	12	11
50%-60%	182	152	49	17	49	46	59
40%-50%	200	108	39	28	142	162	193
30%-40%	214	156	150	111	126	144	127
20%-30%	173	120	105	153	14	18	28
10%-20%	73	14	20	22	3	2	4
0%-10%	10	3	3	2	0	0	4

Tables IX and X show the distribution of the documents across different accuracy ranges using the Levenshtein distance and document similarity metrics for comparing the training data and generated data via the character level language model. These distributions have been reported for evaluating the model with both train and testing data using different input sequence lengths. Longer contexts as input sequences improve the performance of the model which results in higher similarity between the input data and the generated data. As the results in the tables show, training the models with longer input sequences result in better performance of the model and consequently falling more number of generated files in higher accuracy ranges.

TABLE IX
NUMBER OF DOCUMENTS IN DIFFERENT ACCURACY RANGES BASED ON LEVENSHTEIN DISTANCE SIMILARITY METRIC FOR CHARACTER LEVEL LANGUAGE MODEL. CHARS = CHARACTERS

Range \ Input	15 Chars	20 Chars	25 Chars	30 Chars	40 Chars	50 Chars
90%-100%	730	947	972	992	912	1018
80%-90%	113	147	152	155	136	149
70%-80%	107	101	87	96	105	101
60%-70%	157	105	111	127	178	125
50%-60%	145	82	82	76	101	61
40%-50%	148	71	72	41	53	52
30%-40%	112	72	56	50	50	33
20%-30%	27	14	7	2	4	0

TABLE X
NUMBER OF DOCUMENTS IN DIFFERENT ACCURACY RANGES BASED ON DOCUMENT SIMILARITY METRIC FOR CHARACTER LEVEL LANGUAGE MODEL. CHARS = CHARACTERS

Range \ Input	15 Chars	20 Chars	25 Chars	30 Chars	40 Chars	50 Chars
90%-100%	1005	1245	1263	1309	1215	1322
80%-90%	188	118	116	115	174	109
70%-80%	118	48	59	39	67	54
60%-70%	116	67	49	40	44	34
50%-60%	79	39	38	24	25	16
40%-50%	27	22	11	12	11	3
30%-40%	4	0	3	0	3	1
20%-30%	2	0	0	0	0	0

E. Comparing Results with Previous Work

Comparing the distribution of the documents across different accuracy ranges using Levenshtein distance and document similarity as a distance measure in table XI shows that using language models are effective in improving the performance of OCR.

Comparing results of applying language models for OCR post-processing with context based OCR results in [15] show that higher number of documents reside in the accuracy range $90\% - 100\%$ compared to OCR as shown in bold. Moreover, the accuracy improvement of the proposed character and word level language models over OCR is considerable as shown in bold in table XI.

Table XII shows the accuracy improvement of the word level and character level language models for OCR post-processing compared to the results obtained using the pipeline for mixed text recognition in [15] over IAM handwriting database.

TABLE XI
COMPARISON OF NUMBER OF DOCUMENTS IN DIFFERENT ACCURACY RANGES FOR OCR POST-PROCESSING, WRD LVL=WORD LEVEL, CHAR LVL=CHARACTER LEVEL, LEV DST=LEVENSHTEIN DISTANCE, D SIM=DOCUMENT SIMILARITY

	TMIXT [15]		Wrd Lvl		Char Lvl	
Accuracy Ranges	Lev Dst	D Sim	Lev Dst	D Sim	Lev Dst	D Sim
90%-100%	15	45	**998**	**1071**	**1018**	**1322**
80%-90%	496	491	48	92	149	109
70%-80%	861	579	49	26	101	54
60%-70%	147	304	62	16	125	34
50%-60%	14	92	36	49	61	16
40%-50%	4	25	13	142	52	3
30%-40%	0	2	25	126	33	1
20%-30%	0	1	308	14	0	0
10%-20%	0	0	0	3	0	0

TABLE XII
ACCURACY IMPROVEMENT OF LANGUAGE MODEL FOR OCR PRE-PROCESSING OVER IAM HANDWRITING DATABASE. CHAR LVL=CHARACTER LEVEL, WRD LVL=WORD LEVEL,LSV D=LEVENSHTEIN DISTANCE, DOC SIM=DOCUMENT SIMILARITY, PR=PRECISION, IMP=IMPROVEMENT

	TMIXT [15]	This Work		This Work	
Metrics	(%)	Wrd Lvl (%)	Imp (%)	Char Lvl (%)	Imp (%)
PR	65.97	77.31	**11.34**	88.78	**22.81**
Recall	67.31	77.40	**10.09**	85.42	**18.11**
F-score	66.46	77.35	**10.89**	86.87	**20.41**
Lev Dst	77.17	79.42	**2.25**	88.03	**10.86**
Doc Sim	75.06	85.20	**7.14**	95.75	**20.69**

V. CONCLUSIONS

The experiments on IAM handwriting database showed that the word language model have improved the F-score by 10.89% , character level similarity by 2.25% and document level similarity by 7.14%. The character level language model outperformed the word level model and improved the F-score, character level similarity and document level similarity by $20.41\%, 10.86\%$ and 20.69% accordingly. Considering the complexity of the database used for training and also the transcribed files used text prediction, the obtained improvement in OCR accuracy is quite notable. Results show that the character level language model performs better compared to word level language model for improving OCR post processing accuracy on IAM handwriting transcriptions. Improving the OCR results using word level and character level language models also resulted in increasing the number of files falling in accuracy range between $90\% - 100\%$ considerably. The number of files in this accuracy range has increased by $66 - 67$ times using Levenshtein distance comparison and $23 - 29$ times using document similarity metric for Word level and character level language models accordingly.

ACKNOWLEDGEMENT

This work was funded by Applications Service Development Operations Team, Joint Forces Command - Information Systems and Services (ISS), UK.

REFERENCES

[1] Cheriet M, Kharma N, Liu CL, Suen C. "Character recognition systems: a guide for students and practitioners." John Wiley and Sons; 2007 Nov 27.

[2] Bassil Y, Alwani M. "Ocr context-sensitive error correction based on google web 1t 5-gram data set." American Journal of Scientific Research, 2012 Feb 1 (Vol. 50, pp.55-78).

[3] Beaufort R, Mancas-Thillou C. "A weighted finite-state framework for correcting errors in natural scene OCR." International Conference on Document Analysis and Recognition, 2007 Sep 23 (Vol. 2, pp. 889-893).

[4] Broder AZ. "On the resemblance and containment of documents." Compression and Complexity of SEQUENCES, 1997 Jun 13 (pp. 21-29).

[5] Broumandnia A, Shanbehzadeh J, Nourani M. "Segmentation of printed Farsi/Arabic words." International Conference on Computer Systems and Applications, 2007 May 13 (pp. 761-766).

[6] Chang JJ, Chen SD. "The postprocessing of optical character recognition based on statistical noisy channel and language model." Pacific Asia Conference on Language, Information and Computation, 1995 Dec (pp. 127-132).

[7] Doush IA, Alkhateeb F, Gharaibeh AH. " A novel Arabic OCR post-processing using rule-based and word context techniques." International Journal on Document Analysis and Recognition, 2018 Jun 1, (Vol. 21, No. 1-2, pp. 77-89).

[8] Perez-Cortes JC, Amengual JC, Arlandis J, Llobet R. "Stochastic error-correcting parsing for OCR post-processing." International Conference on Pattern Recognition, 2000 Sep 3 (Vol. 4, pp. 405–408).

[9] Strohmaier CM, Ringlstetter C, Schulz KU, Mihov S. "Lexical postcorrection of OCR-results: The web as a dynamic secondary dictionary?." International Conference on Document Analysis and Recognition, 2003 Aug 6 (pp. 1133-1137).

[10] Zhuang L, Zhu X. "An OCR post-processing approach based on multi-knowledge." International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, 2005 Sep 14 (pp. 346-352).

[11] Mikolov T, Karafit M, Burget L, ernock J, Khudanpur S. "Recurrent neural network based language model." Annual conference of the international speech communication association 2010.

[12] Kiros R, Salakhutdinov R, Zemel R. "Multimodal neural language models." International Conference on Machine Learning, 2014 Jan 27 (pp. 595-603).

[13] Kissos I, Dershowitz N. "Image and text correction using language models." International Workshop on Arabic Script Analysis and Recognition, 2017 Apr 3 (pp. 158-162).

[14] Bryant C, Briscoe T. "Language model based grammatical error correction without annotated training data." Workshop on Innovative Use of NLP for Building Educational Applications, 2018 Jun (pp. 247-253).

[15] Medhat F, Mohammadi M, Jaf S, Willcocks CG, Breckon TP, Matthews P, McGough AS, Theodoropoulos G, Obara B. "TMIXT: A process flow for Transcribing MIXed handwritten and machine-printed Text." International Conference on Big Data, 2018 Dec 10 (pp. 2986-2994).

[16] Marti UV, Bunke H. "The IAM-database: an English sentence database for offline handwriting recognition." International Journal on Document Analysis and Recognition, 2002 Nov (Vol. 5, No. 1, pp. 39-46).

[17] Karpathy A. "The unreasonable effectiveness of recurrent neural networks." http://karpathy.github.io/, Accessed: 2019-03-20.

[18] Hochreiter S, Schmidhuber J. "Long short-term memory." Neural computation. 1997 Nov 15; (Vol. 9, No. 8, pp. 1735–1780).

[19] Loper E, Bird S. "NLTK: the natural language toolkit." Association for Computational Linguistics on Interactive poster and demonstration sessions, July 2004, (pp. 69-72).

[20] Brownlee J. "Machine learning mastery." https://machinelearningmastery.com/, Accessed: 2019-03-20.

[21] Vladimir I Levenshtein. "Binary codes capable of correcting deletions, insertions, and reversals." Soviet physics doklady, 1966 Feb 10 (Vol. 10, No. 8, pp. 707-710).

[22] Salton G, Wong A, Yang CS. "A vector space model for automatic indexing." Communications of the ACM. 1975 Nov, (Vol. 18, No. 11, pp. 613-620).

[23] Pagliardini M, Gupta P, Jaggi M. "Unsupervised learning of sentence embeddings using compositional n-gram features", Transactions of the Association for Computational Linguistics, 2018 , (Vo. 1, pp. 528-540).

[24] Fady Medhat, “mixed-iamdb“ https://bitbucket.org/DBIL/mixed-iamdb, Accessed: 2019-03-20.
[25] Marti UV, Bunke H. “A full English sentence database for off-line handwriting recognition.“ International Conference on Document Analysis and Recognition, 1999 Sep 22 (pp. 705-708).