Post-OCR text correction using multi-task learning

Magdalena Bilska (s4086511), Ivo de Jong (s3174034) & Travis Hammond (s2880024)

Abstract

Optical Character Recognition (OCR) is a widely used framework of digitizing both handwritten and printed text. However, due to low quality of materials, the results are often non-ideal (Rigaud et al., 2019). To correct the errors made by OCR software, a separate task consisting of error detection and editing can be performed, so-called post-OCR text correction. Due to the high relatedness of the two sub-tasks, a promising approach is that of Multi Task Learning (Caruana, 1997), in which a shared neural network is used to improve the performance on two or more related tasks. To investigate the potential of use of MTL for the purpose of post-OCR correction, a Bi-directional Long Short Term Memory (BLSTM) network is used. Although the experimentation shows no improvement in performance of the MTL model as compared to the classic sequential approach, the results show a promising trend and several conclusions are drawn.

1 Introduction

Optical Character Recognition (OCR) is a system of converting images of printed, handwritten, or typed text into their machine-encoded version (Schantz, 1982). This technology has a wide spectrum of applications, ranging from digitizing mundane documents such as scanned contracts to archiving corrupted historic data. However, especially in case of the latter, even state-of-the-art methods often yield suboptimal results; this is usually due to poor quality of documents (Rigaud et al., 2019). Moreover, digital libraries are oftentimes "shelve" resources processed with outdated OCR technologies (Rigaud et al., 2019). In order to deal with these issues, the task of post-OCR text correction has emerged and established itself as a popular and important sub-task in language technology (Rigaud et al., 2019). After all, manually annotating vast amounts of different types of documents (e.g. medieval manuscripts with various fonts, newspaper articles from specific geographic regions, etc.), or re-recognizing entire texts with better software is a very tedious and time-consuming challenge. Comparatively, it seems simpler to develop a model that deals with post-OCR data and corrects it.

Multi-task learning (MTL) (Caruana, 1997) with deep neural networks (NNs) is a practice which recently gained notable momentum within the field of Natural Language Processing (NLP). Although the main purpose of MTL is using information utilized in two or more related NLP tasks to enhance the generalizability for all of them (Zhang and Yang, 2017), MTL has also been shown to be useful in dealing with scarcity of training data (Li et al., 2019). In its most popular form, the MTL framework assumes sharing the hidden layers of the NN among tasks, with each task having its own output layer (Li et al., 2019).

In this paper, an attempt of building such a simple architecture using a Bidirectional Long-Short Term Memory (BLSTM) network will be captured (Hochreiter and Schmidhuber, 1997). In particular, the performance of a MTL BLSTM will be compared to the performance of its sequential variant - here called Single Task Learning (STL) BLSTM. Specifically, the comparison of these two variants will be made using the two sub-tasks of post-OCR text correction: detection and correction (Rigaud et al., 2019). The MTL network is expected to show superior performance compared to the classic STL framework.

2 Related Work

Recently, MTL with deep NNs gained significant attention, especially in the field of NLP where it has shown promising results on various tasks. The MTL methods can be divided into two kinds, called hard and soft parameter sharing (Li et al., 2019). The former can be seen as rather straightforward. Simply put, it encompasses the use of a single NN for the purpose of two or more tasks. While the hidden layers are shared between all of them, each task is assigned its own output layer (Li et al., 2019). As opposed to that, in soft parameter sharing each task is assigned a NN of its own. The MTL in such networks results from regularization that takes place between them (Li et al., 2019). Due to both its simplicity and performance, hard parameter sharing seems to be the preferred method in the field of NLP (Li et al., 2019). MTL networks can consist of very complex architectures and be used for as many as four complex tasks (Sanh et al., 2019). They have also been found to improve the overall performance in NLP tasks Li et al. (2019). For example, MLT resulted in an improvement in the task of named entity recognition Aguilar et al. (2019). The authors used three kinds of features: characters, words, and lexical characteristics. Each of them was paired with a separate network for their encoding: a Convolutional Neural Network (CNN), a BLSTM, and gazetteers, respectively. These encodings were further concatenated and passed through a common dense layer (Aguilar et al., 2019). The network of Aguilar et al. (2019) can be seen as an extension of the work of Limsopatham and Collier (2016), who used a simpler BLSTM for the sake task. Inspired by this line of research, this paper will present a much more simplified version of these approaches and apply it to a different task - that of post-OCR correction.

3 Model

For the purpose of the project, the PyTorch environment was used (Paszke et al., 2019). Entirety of the used code can be accessed under this link: GitHub repository.

The MTL model consisted of a BLSTM with a hidden layer of 32 neurons that acts as a shared input layer, followed by two distinct output layers, one per each task. A fully connected layer with a single sigmoidal neuron was used to classify the current character as corrupted or not, while another fully connected layer with 27 neurons (26 lowercase characters and the space) and a Softmax activation was deployed to predict the correct one-hot-encoded character. Hence, for a given sequence, the MTL model determined whether the middle character is corrupted, as well as replaced that character with its prediction.

On the other hand, the STL setting consisted of two distinct NNs. In one, a BLSTM input layer with 32 hidden neurons and a fully connected output layer with a single sigmoidally activated neuron classified a given character as corrupted or not. In the other, an identical BLSTM was connected to a fully connected layer with 27 neurons and a softmax activation. If the first NN classified the middle character of a sequence as corrupted, the second one replaced that character with a prediction.

Binary cross-entropy (BCE) loss was used to determine the error in both models. The learning rate was scheduled using the Adam(Kingma and Ba, 2014) optimizer.

Figure 1 highlights the differences between the two models.

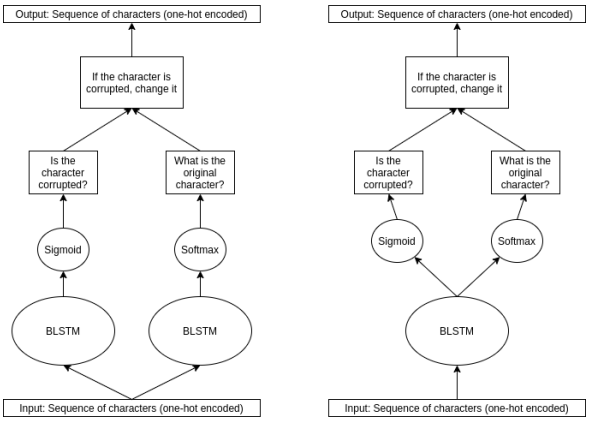


Figure 1: The STL model (left) utilizes a separate BLSTM for each task, while the MTL model (right) uses a single BLSTM.

4 Experiments

Data

Pre-processing The WikiText-2 corpus (Merity et al., 2016) was used for experimentation. This dataset contains tokens of words, as well as non-words. Firstly, since not all characters are neccesarily ASCII, all characters are first converted to the nearest ASCCI. Due to the fact that the dataset is available as a tokenized list, all the non-

alphabetical tokens can easily be discarded. The remaining tokens were concatenated into a single string with spaces separating the words. Finally, the entire text was converted into lower case.

The decision to simplify the input in this manner was made due to the fact that allowing for punctuation would dramatically increase the complexity of the task. This is because processing the complex input would require a deeper, more semantic understanding of a sentence to, for example, be able to distinguish whether the sentence should end with an exclamation mark or a question mark. If only lowercase alphabetical characters are allowed, the problem can be mostly reduced to the syntactic level, requiring the model to learn only what words look like and not what they mean.

After pre-processing, a copy of the text was made and altered by first determining whether a character should be corrupted (30% chance), and if so, substituting it for a random character drawn from a uniform distribution. These two versions (original and corrupted) of the text were used as a labeled OCR dataset.

The data was first sliced into sliding-window sequences of 21 characters with a stride of 1, which were subsequently divided again. The result was 4.5 thousand batches of 2048 sequences ready to be fed to the model.

An attempt to run the models on real-life OCR data was made Molla and Cassidy (2017). However, due to sole use of characters as features, it was unsuccessful. This is likely due to the fact that real OCR data oftentimes misses whole characters, which are not recognized and skipped. This phenomenon lead to the models displacing the reading frame, making them unable to learn anything meaningful.

Evaluation Due to the large amount of data available, the dataset was split into training, validation and development sets. Training and development sets each contained 9.4 million- while the validation set contained 10.4 million sequences.

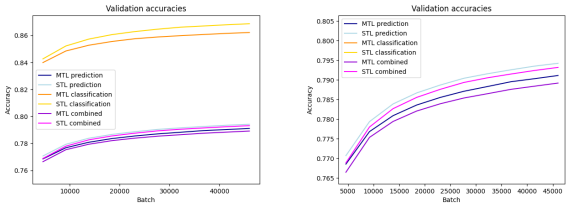
Following the training phase, the performance of both models was measured on the validation dataset, using accuracy as a metric. Both models were run for 10 epochs of 46k batches.

For the purpose of this project, accuracy is defined as the percentage of output characters that match their labels. It can be calculated for each of the sub-tasks, using a very straightforward equation:

$$accuracy = \frac{number\ of\ correct\ predictions}{total\ number\ of\ predictions}$$

	Classification	Prediction	Combined
MTL	0.8619	0.7894	0.7876
STL	0.8690	0.7930	0.7915

Table 1: The validation accuracies of both models.



(a) Development accuracies (b) Zoomed in

Figure 2: Development accuracies for the MTL and STL models (left), and a zoomed in view of the prediction and combined accuracies (right).

5 Results

Validation accuracies of both models can be seen in Table 1. The accuracies are decomposed into prediction, classification and the combined (true) accuracy. These predictions reflect the performance on the error correction and detection tasks, respectively. The combined accuracy is the true measure of whether or not the output character matched its label.

Figure 2 shows the accuracies over time. The classification accuracy for both models appears to be higher than the combined or prediction accuracies. This is expected, since classification is a relatively easy task as compared to prediction. Another observation is that the combined accuracy is always slightly lower than prediction due to the interdependence between them. Both models make a change based on their prediction output only if their classification output indicates a corruption.

We can see from Figure 2 and Table 1 that the single task learning model performs better than the multi task model across all of the decomposed terms. This is somewhat surprising, as the MTL model was expected to benefit from the added regularization Zhang and Yang (2017). However, the lack of a difference between models can be attributed to the large size of the used dataset, which prevented overfitting in the STL model.

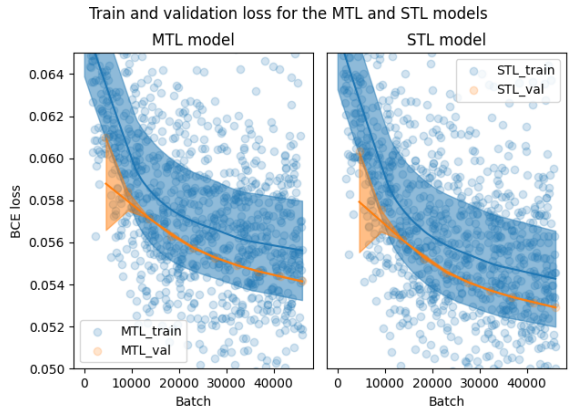


Figure 3: Binary cross-entropy losses during training (blue) and development (orange) for both models.

	Classification	Prediction	Combined
MTL	0.7523	0.6474	0.6969
STL	0.7397	0.647	0.6951

Table 2: The validation accuracies of both models, with 100 batches of size 100 running for 40 epochs.

The losses in Figure 3 behave as expected considering the short training time. The smoothed lines, as well as the error lines, were calculated using LOWESS scatter plot smoothing. It can be expected that after more epochs the training loss would improve beyond the validation loss and the model would start overfitting on the data. That is when the MTL model, due to its regularization effects, would start outperforming STL.

To verify the hypothesis presented in the previous paragraph, the experiment was re-run using a smaller dataset; the number of batches was capped at 100, and the batchsize was set to 100. At these settings, the models were run for 40 epochs. As can be seen in Figures 4 and 5, as well as Table 2 this time the MTL model did marginally better as compared to STL. The difference is small, however, so further experiments would have to be made to establish the significance of this difference.

6 Conclusions and Future Work

The results can be seen as contradictory to expected. Seemingly, the additional context shared between the tasks in form of parameters did not contribute to a significant difference in the mod-

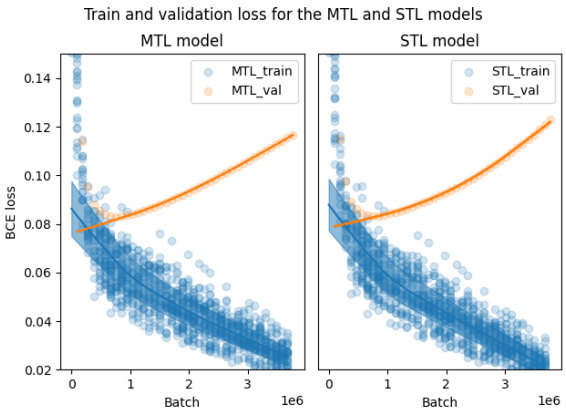


Figure 4: Binary cross-entropy losses during training (blue) and validation (orange) for both models, with 100 batches of size 100 running for 40 epochs.

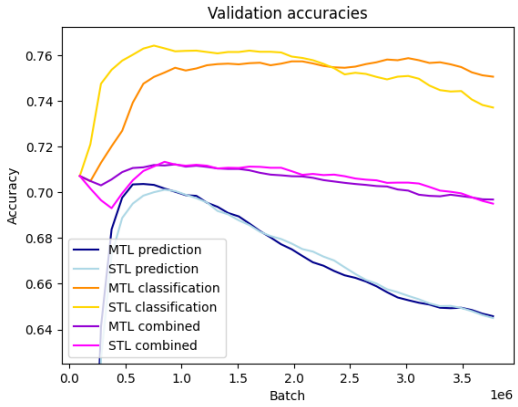


Figure 5: Validation accuracies for the MTL and STL models, with 100 batches of size 100 running for 40 epochs.

els' performance (Table 1). At the first attempt, the STL network actually performed worse than MTL. However, this finding is likely due to several factors: initial large size of the dataset and insufficient training. It is likely that upon further experimentation, the expected differences between the accuracies of the models would emerge. After all, an additional run using a smaller-size set resulted in the two models yielding almost identical results (Table 2.

Additionally, the experiment suffered from numerous shortcomings. First of all, the used dataset was created as opposed to drawn from real-life. Second of all, the data was only analysed on a single level - that of a character. Moreover, the task picked turned out to come with an extremely high sensitivity to missing data.

In order to improve the results, these limitations should be improved upon. Using a realistic OCR dataset might introduce specific patterns which the created data did not capture; several features could be taken into account in order to increase the amount of information (such varying features can be then encoded and concatenated, as per Aguilar et al. (2019)); a creative way to deal with the issue of reading frame misplacement could be implemented.

Contributions

Ivo de Jong: finalized both models and implemented the classifier; implemented validation and testing; implemented logging and graphing and revised the data generator; trained and tested the model with the full dataset; added some small changes to the report concerning references and the pre-processing methodology
Travis Hammond: composed a skeleton code with the necessary method stubs; made the first draft of a model and the training loop; re-ran the experiment with the subset of data and updated the graphs; wrote drafts of Model, Experiments and Results sections of the report
Magdalena Bilska: wrote the Abstract, Introduction, Related Work, and Conclusions and Future Work sections of the report; revised the Model, Experiments, and Results sections; restructured and edited the whole report

References

Aguilar, G., S. Maharjan, A. P. López-Monroy, and T. Solorio (2019). A multi-task approach for named entity recognition in social media data. *arXiv preprint arXiv:1906.04135*.

Caruana, R. (1997). Multitask learning. *Machine learning 28*(1), 41–75.

Hochreiter, S. and J. Schmidhuber (1997, 12). Long short-term memory. *Neural computation 9*, 1735–80.

Kingma, D. P. and J. Ba (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Li, J., X. Liu, W. Yin, M. Yang, and L. Ma (2019). An empirical evaluation of multi-task learning in deep neural networks for natural language processing. *arXiv preprint arXiv:1908.07820*.

Limsopatham, N. and N. Collier (2016, December). Bidirectional LSTM for named entity recognition in twitter messages. In *Proceedings of the 2nd Workshop on Noisy User-generated Text (WNUT)*, Osaka, Japan, pp. 145–152. The COLING 2016 Organizing Committee.

Merity, S., C. Xiong, J. Bradbury, and R. Socher (2016). Pointer sentinel mixture models. *CoRR abs/1609.07843*.

Molla, D. and S. Cassidy (2017). Overview of the 2017 alta shared task: Correcting ocr errors. In *Proceedings of the Australasian Language Technology Association Workshop 2017*, pp. 115–118.

Paszke, A., S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035.

Rigaud, C., A. Doucet, M. Coustaty, and J.-P. Moreux (2019). Icdar 2019 competition on post-ocr text correction.

Sanh, V., T. Wolf, and S. Ruder (2019). A hierarchical multi-task approach for learning embeddings from semantic tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Volume 33, pp. 6949–6956.

Schantz, H. F. (1982). *History of OCR, optical character recognition*. Recognition Technologies Users Association.

Zhang, Y. and Q. Yang (2017). A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*.