EasyChair Preprint

№ 992

Feedback Learning: Automating the Process of Correcting and Completing the Extracted Information

Rakshith Bymana Ponnappa, Khurram Azeem Hashmi,
Syed Saqib Bukhari and Andreas Dengel

May 12, 2019

Feedback Learning: Automating the Process of Correcting and Completing the Extracted Information

***Abstract*—In recent years, with the increasing usage of digital media and advancements in deep learning architectures, most of the paper-based documents have been revolutionized into digital versions. These advancements have helped the state-of-the-art Optical Character Recognition (OCR) and digital mailroom technologies become progressively efficient. Commercially, there already exists end to end systems which use OCR and digital mailroom technologies for extracting relevant information from financial documents such as invoices. However, there is plenty of room for improvement in terms of automating and correcting post information extracted errors. This paper describes the user-involved, self-correction concept based on the sequence to sequence Neural Machine Translation (NMT) as applied to rectify the incorrectness in the results of the information extraction. Even though many efficient Post-OCR error rectification methods have been introduced in the recent past to improve the quality of digitized documents, they are still imperfect and demand improvement in the area of context-based error correction specifically for the documents involving sensitive information. This paper further illustrates the capability of sequence learning with the help of feedback provided during each cycle of training, yields relatively better results and have outsmarted the state-of-the-art OCR error correction methods.**

***Index Terms*—Document Understanding, Post IE Error Correction and Completeness, Sequence to Sequence Neural Machine Translation**

I. Introduction

From the very beginning of computers to the current digital era, computers have been remarkably transformed and upgraded. Among many applications, the main purpose of computer is to process the information. In this age of automation, where the primary aim is to digitize every thing, one of the essential task is to capture and process all kinds of documents. While the documents containing texts are rather interpretable but working on the scanned images of the documents is certainly not an effortless operation.

Digital Mailroom System is defined as the system which receives the mails and processes them automatically. The processing part in the mailroom system includes various functionalities like digitizing the mail, it's classification, distribution among the related personnel and so on. Optical Character Recognition (OCR) [7] is the process that allows us to convert documents such as scanned paper images, photographs or invoices into machine readable and editable information by applying various image processing methodologies [18], [5]. These OCR systems have been used mainly to extract handwritten or typed information in the digital mailroom systems. The problem arises when these information extraction systems cannot extract the exact information due to many reasons like the source graphical document itself is not readable, scanner has a poor result, characters in the document are too close to each other resulting in problems like reading "Ouery" instead of "Query", "8" instead of "3", to name a few. It is even more challenging to correct errors in proper nouns like names, addresses and also in numerical values such as telephone number, insurance number and so on. An example of a sample invoice can be seen in Figure 1. Let's suppose we need to extract first name, last name, date of birth, insurance number and hospital name from this invoice image, then some of the possible errors in the extracted information along with the respective ground truth is shown in Figure 2.

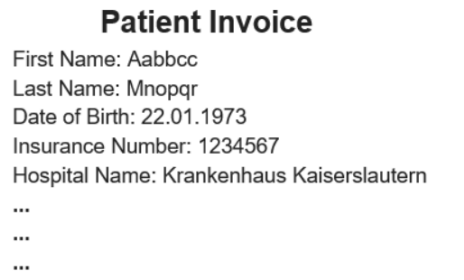

Fig. 1: Synthetic sample invoice image containing information of the patient.

Extracted Information:
A@bboc Mn0pqr 22.01.1978 1284567 Krank€nhau$ Kajserslaut€nn

Ground Truth:
Aabbcc Mnopqr 22.01.1973 1234567 Krankenhaus Kaiserslautern

Fig. 2: Extracted information from the synthetic sample invoice (as shown in Figure 1) and it's corresponding ground truth. Some of the errors in the extracted information can be observed here.

Hence these erroneous words can simply affect the processing of data specifically when the data is important such as personal information of clients for a company. Hence, these problems lead to manual labor where a person has to read all the extracted information, distinguish the errors and correct the mistakes every time. Our approach stands to overcome this human labor

part which shall reduce the human effort involved in correcting and completing the errors and missing information from the extracted data.

A considerable amount of research has been conducted in the area of post-Information Extraction (IE) corrections with various techniques. Few of those techniques involve machine learning algorithms to rectify the textual errors obtained from OCR [11],[12],[10] while in another approach, they propose the method to select most suitable correction among all of the available options [13]. It has also been clearly illustrated that the errors generated from the IE systems are more diverse than the handwriting errors [6] and [9]. In one of the previous works [14], NMT is already used in correcting post-OCR errors in the historical documents.

In this paper, the data on which we worked on consists of customer and company profiles based on the type of invoice processed. For example, in a health care invoice the data might have all the personal information such as *First name*, *Last name*, *Date of birth*, *Hospital Address*, *Type of Medicine* and so on. We have 2 cases for this data, one we generate ourselves and call it as *Synthetic data use case* whereas another one is a *Private data use case* from a health care insurance company. One of the biggest problem in these use cases is that we cannot take much help from any language model because most of the values in the data are the proper nouns. To overcome this problem we have introduced a new technique called *Feedback Learning* over sequence to sequence learning technique.

The rest of the paper is organized as follows. Section II explains the working of NMT whereas Section III describes the Feedback Learning process and Section IV defines the methodology used in detail. Section V illustrates the design and experiments. Section VI discusses the evaluation of the obtained results while Section VII concludes the paper.

II. Sequence to Sequence Neural Machine Translation

Neural Machine Translation (NMT) is a machine translation method that uses deep neural networks. Back in the old days, the usual method for the sentence-based translation system is executed by dividing the source sentences into multiple pieces and then translated them sentence-by-sentence, but this leads to severe problems in the translation outputs because of the many reasons like less fluency in the language, no or very less context awareness, order of the translated sentence is totally inaccurate according to the grammatical rules of the objective language and so on. Lately, the sequence-to-sequence models [17] [1] have done wonders not only in such kind of problems but also in speech recognition and text summarization.

NMT is based on these sequence-to-sequence models which basically mimics how human interprets any sentence. We humans read the entire sentence, interpret the meaning, and then map those words into respective translation. Same is the case regarding working of NMT [14].

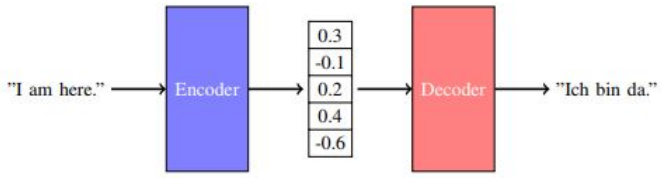

Fig. 3: A simple representation of an encoder-decoder architecture [14] which translates the English sentence "I am here" into the German language.

The Figure [3] explains the workflow of NMT. First, the encoder converts a source sentence into a "meaning" vector (array of values) which is basically the latent representation of the features. These features will be passed through a decoder and the decoder network tries to learn them accurately, having the vocabulary and grammar of the target language, it produces a translation.

The implementation of these decoders and encoders have been executed using different deep learning architectures but since we are dealing with sequential data (one line per profile), The Recurrent Neural Network (RNN) is the most suited to work with encoder and decoder. The encoder in our model is uni-directional LSTM whereas the decoder is AttentionalRNN. We selected LSTM because of its outstanding performance in the fields of speech recognition, language modeling, translation, and image captioning.

III. Feedback Learning

In this paper, we define the feedback learning as a learning cycle in which the network is being trained by continuously receiving an input from the user and it will learn those patterns of correction and after some point in time, when the network is properly trained it will start resolving those errors automatically without the external help from the user. To make it simpler, in the first feedback cycle, network after completing the first training will translate the results, and those results will be incorporated along with the training data in the second feedback cycle. This approach benefits profoundly in the health-care sector as most of the information in a patient's billing invoice is important and any missing or inaccurate report can cause complication. With the objective of decreasing human involvement, we focus on achieving a system that auto-corrects the errors after the feedback cycle. So whenever the network encounters similar faults or misplaced information in the data, it predicts the appropriate output by keeping track of the context. To implement this concept, we have used Neural Machine Translation (NMT) that has been applied earlier in the past as well for post-OCR correction [14].

IV. Methodology

In this section, we present the use of an open source library called OpenNMT [8] which uses the neural sequence modeling

to help improving the output of information extraction. We compare our network model performance with the baseline method of using the basic Hunspell [15] German dictionary look-up to prove our point that the feedback Learning is not a normal spell correction method.

A. Word Based Sequence-to-Sequence model

In this model, we use encoder-decoder architecture by treating post-IE data as neural sequence translation problem. The model is based on a word level tokenization, the encoder considers each of the sentence as a sequence of words. The configuration of our network is explained as below:

1) Layers : 2 (uni-directional RNN LSTM encoder and Attentional RNN LSTM decoder)
2) Size of Layers : 512
3) Word Embedding Size : 512
4) Dropout Percentage : 30
5) Optimizer : Adam Optimizer
6) Learning Rate : 0.001
7) Beam Width : 4
8) Batch Size : 32

Here, we divide our problem into three sub-tasks *Preprocessing*, *Training* and *Inference*.

1) Preprocessing

We consider the dataset generated using Faker [3] as described in Section V-A1 and the very first step is to build source and target vocabularies by specifying the size of vocabulary. The data consists of parallel source (src) and target (trg) data with one sentence per line and each of the fields are separated by spaces. Each line in the source file corresponds to the equivalent line in the target file. The source file consists of erroneous data from the information extraction and the target file consists of correct data which acts as ground truth. It indicates that the error data (src) has to be translated into correct data.

2) Training

From the dataset, we prepared the configuration described above is saved in the config file to train and evaluate in parallel on our dataset. Validation in parallel helps in evaluating our model convergence during the training. The training is done on a CUDA [2] enabled NVIDIA GPU and the log information of training is written to Tensorboard [19], which helps us to visualize and monitor training and evaluation loss and few other characteristics of our model like learning rate and data distribution.

3) Inference

Once the training is completed, we have our saved model to evaluate the performance of the model architecture against other baseline methods. This saved model is now used to predict and correct the mistakes from our test dataset. The predictions are done by using beam [19] search where multiple hypothetical target predictions are considered for each sequence during the individual step and most relevant prediction is taken.

B. Dictionary look-up using Hunspell

In this method, we tokenize the test dataset and pass each of the words to our custom dictionary using Hunspell[15] to correct the mistakes and select the best prediction. Here we use our own dictionary because the dataset in our experiment contains proper nouns and using a generic German dictionary would produce a bad result by default. Since our dataset has numerical values such as insurance number and date of birth, we correct this information using regular expression because any dictionary would not be able to predict the numerical mistakes.

V. Experimental Design

In this section, we present the different kinds of datasets used for training and testing our network model. We discuss how each of the datasets is generated and give details of the test cases involved in the translation of the output. After the prediction, we evaluate the resulting output from our different test cases and analyze the performance of the model later.

A. Datasets

1) Synthetic Use Case

This dataset is generated from an open source Python library named Faker [3]. We considered private dataset as a base and tried to replicate a similar format of data by generating data from the library. For this experiment, we created synthetic data having 150,000 user profiles with the corresponding fields *(FirstName, LastName, Address, Hospital Name, Hospital Address, Sex, Date of Birth, Phone Number, Insurance Number)*. In this data, there are 25,000 unique user profiles and the rest 125,000 are the replication from the unique profiles with different combinations of data fields mentioned above.
Now that we have ground truth, the challenge was to simulate erroneous data almost identical to OCR output. For this purpose, we used a document analysis tool OCRopus [16] to identify the statistics of common information extraction errors in a real-world scenario. From the character distribution stats through OCRopus, we generated error data by replacing certain characters from the ground truth with the identified errors. For example ('a' : 'o', 'e':'€'). We created artificial noise by corrupting 95% of the data generated using Faker with character replacements. We have assigned percentages to each of the particular characters which need to be replaced. By the end of this process, there will be two datasets, one with ground truth and the other with an error data. For a real world scenario, the erroneous data represents the extracted information coming out from a digital mailroom system, and the ground truth data represents the manually corrected information with the help of human verification. The sample information sequence,

ground truth and the respective error profile are mentioned below. It is important to note that the delimiter between the two consecutive information is space, however, it is also possible that space could occur within the information element, for example, the address might have multiple information such as street name, postcode, city, and country (Leonid-Renner-Platz 71165 Wurzen Hamburg Germany) in the sample.

Information sequence in our dataset follow this order **<Address, Birthdate, Blood Group, First Name, Insurance Number, Hospital Address, Hospital ID, Hospital Name, Hospital City, Hospital Postcode, Last Name, Phone Number, Gender>**

Ground Truth : *Leonid-Renner-Platz 71165 Wurzen Hamburg Germany 2004-06-08 A+ Reimar 422893598198 Sankt Annen Str.9 990702828 Krankenhaus St. Anna-Stift Löningen 49624 Wilms 03549 58413 M*

Error Data: *L€onjd-R€nn€r-Platz 71165 Wurz€n Hamburg, G€rmony 2004-06-0O A+ Reimar 0422893598198 $ankt Amen Str.9 960702328 Krankenhau$ St. Anna-Stift Löningen 49624 Wilm$ 03549 58413 M*

The dataset is divided into train, test and validation sets having two parallel documents (Ground truth and error data) since OpenNMT requires a source(error data) and target(ground truth) as an input during training. The distribution of the data into Training, Validation, and Testing is elaborated in the Table I.

2) Private Use Case

This dataset is rather a small one as compared to synthetic but it is based on the actual information from an insurance company which makes this a critical use case. Since the number of given unique profiles is only 94 which are certainly not enough to train a deep neural network so we increased the number of profiles by augmenting the data and introducing different errors in a similar way as we have made in the synthetic data use case. This approach leads to 20,000 profiles where each profile is treated in a single sentence and we have corrupted 95% of data in the same manner as we did for the synthetic data use case. The distribution of the data into Training, Validation, and Testing for this case is also explained in Table I.

Dataset Statistics for both of the use cases				
Datasets	# Sentences	Training	Validation	Test
Synthetic	150,000	70,000	10,000	30,000
Private	20,000	12,000	1,000	3,000

TABLE I: Data distribution showing the number of samples used in each case for the first feedback learning cycle where one sample is a single user profile.

The Table II illustrates the data distribution of feedback cycle 2 which is basically the second iteration. In this loop, we combine the prediction results obtained from the feedback cycle 1 with our training data. Hence, the size of the training set for synthetic use will be 100,000 samples (70,000 + 30,000 from feedback cycle 1). After updating the vocabularies in the preprocessing stage, we train the model again for further 10,000 steps in the synthetic use case and 5,000 more steps in private use case. While deciding the number of steps in the second feedback learning cycle, the evaluation loss is considered as a significant factor. Once the model is trained, we infer the test set from Table II on both of our models and calculate the prediction accuracy.

Dataset Statistics for both of the use cases				
Datasets	# Sentences	Training	Validation	Test
Synthetic	150,000	100,000	10,000	30,000
Private	20,000	15,000	1,000	3,000

TABLE II: Data distribution showing the number of samples used in each case for the second feedback learning cycle where one sample is a single user profile.

VI. Performance Evaluation

In this section, we present the results of the various test cases. This helps us to understand whether the predicted output of our deep learning trained network has improved or not. The accuracy is defined as how identical is the predicted document with respect to the target document. To calculate the accuracy we use Levenshtein distance[4] as it helps in identifying the number of edit operations required to transform the error data to ground truth and it gives the similarity measure between two documents. In all of our test cases, we have used token level error measurement. The following test cases were used :

- **Test Case 1** : In this experiment, we used the test set that was obtained by splitting the erroneous data.
- **Test Case 2** : In this experiment, we introduced a new set of errors to test case 1. We did this to check if the model still predicts the output sequence correctly even when it has not seen or trained on new kind of errors.
  Example *: Lconid-Rcnncr-Platz 77165 Wurzen Hamburg Gcrmany 2oo4-06-08 A+ Reiiimar 0422893598198 Sankt Annen Str.9 990702828 Krankenhaus St.Anna-Stift Leaningen 49624 Wiiilms 03549 58413 M*
- **Test Case 3** : In this experiment, we removed a few data fields from the ground truth that was generated initially.
  Example *: Leonid-Renner-Platz 71165 Wurzen Hamburg Germany 2004-06-08 A+ Sankt Annen Str.9 990702828 Krankenhaus St. Anna-Stift Löningen 49624 Wilms 03549 58413 M*
- **Test Case 4** : In this experiment, we removed one random field on even line and two random fields if it's an odd line in the dataset.
  Example *: L€onjd-R€nn€r-Platz 71165 Hamburg G€rmany 2004-06-0O A+ Reimar 0422893598198 $ankt Amen Str.9 Krankenhau$ St.Anna-Stift Löningen 49624 Wilm$ 03549 58413 M*
- **Hunspell Case 5** : This experiment is followed on the basis of the second approach mentioned in the

methodology (IV-B) of using Hunspell[15] to check the prediction.

A. Synthetic Use Case

The Table III explains the results for the individual test cases which were elaborated earlier and it can be clearly seen that our model has predicted better results in all of the first four test cases. Another important thing to shed light on is the difference between the performances from the first four cases and the fifth case which is the Hunspell. This clearly demonstrates that the feedback learning technique has absolutely outsmarted the use of the dictionary in post-IE correction specially when the dataset consists of many proper nouns while the Table IV presents the results of Table III showing the improvement in accuracy after the second feedback learning cycle.

Test Cases	Accuracy Before	Accuracy After
1	27.14	93.21
2	27.03	92.47
3	38.80	91.48
4	21.08	90.89
5	27.14	31.39

TABLE III: Accuracy in each of the test case before and after applying the trained model on extracted information in the first feedback learning cycle.

Test Cases	Accuracy Before	Feedback Accuracy
1	93.21	94.91
2	92.47	93.22
3	91.48	93.18
4	90.89	92.63
5	31.39	32.12

TABLE IV: Accuracy in each test case before and after applying the trained model on extracted information in the second feedback learning cycle.

The Figure [4] displays the evaluation loss of our training model. The training has been stopped by us after 15,000 steps since the evaluation loss starts ascending and could cause the model to overfit. The orange line indicates the training loss while the blue line implies the evaluation loss.

B. Private Use Case

For the private use case, we have also achieved fairly good results. The Table V describes the result on post OCR information extraction done on the data directly taken from the OCR output.

The Figure [5] is the evaluation loss of private use case and the training is done only until 10,000 steps as the dataset is small and it has least validation loss at that particular point.

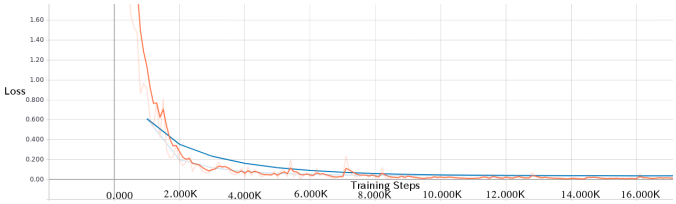

Fig. 4: Training progress of the network after every 2,000 steps on Synthetic use case to monitor the evaluation loss. The model starts to converge after 15,000 steps.

Test Cases	Accuracy Before	Feedback Accuracy
1	59.45	**91.15**

TABLE V: Accuracy of Private test case before and after applying the trained model on extracted information in the first feedback learning cycle.

The Table VI represents the results of Table V and it clearly depicts the improvement in accuracy after the second feedback learning cycle in the private use case as well.

Fig. 5: Training progress of the network after every 1,000 steps on Private use case to monitor the evaluation loss. The model starts to converge after 9,000 steps

Test Cases	Accuracy Before	Feedback Accuracy
1	91.15	**92.61**

TABLE VI: Accuracy of Private test case before and after applying the trained model on extracted information in the second feedback learning cycle.

C. Significance Test

We wanted to prove that the new model trained in the second feedback cycle having the prediction results from the first feedback cycle would give us better results as compared to the model trained in the first feedback cycle. We proposed a null hypothesis which is precisely the contradiction of our assumption. In order to reject or accept the null hypothesis, we performed the significance test by applying reinforcement sampling technique and divided our synthetic test set of 30,000 samples into 10 pieces of 3,000 samples each and a private test set of 3,000 samples into 10 batches of 300 samples. We calculated the prediction accuracy by inferring each of these samples and our t-test value identifies that the new model produced after second feedback cycle has a significant

improvement in performance and predicted better results. This also proves our assumption and rejects the null hypothesis. It also helps us recognizing another point that retraining with the predicted output will always improve the results each time accounting to the feedback patterns and reduces the human involvement in correcting the errors with the increase in the size of the dataset.

VII. Conclusion

Post-IE error corrections have become a vital step for processing information from the graphical documents. Sometimes these corrections develop into a challenging task, specially when the data is not prominent in the scanned images, along with it, when we are treating proper noun error correction. In this paper, we proposed the new feedback learning technique that explains how erroneous words after information extraction can be corrected by reducing the human effort in the detection and correction of post-IE errors. We have implemented this concept through deep learning architecture using OpenNMT which is an open source tool. Our method manages to convert 27.14% accuracy of information extraction in test case 1 into 93.21% accuracy of information extraction after first feedback learning cycle and this accuracy is further increased up to 94.91% in the second feedback learning cycle in the synthetic use case. While in the private use case it, converts 59.45% accuracy of information extraction for the test case 1 into 91.15% accuracy after first feedback learning cycle which is further improved to 92.61% in the second feedback learning cycle.

Our current results have involved word based tokenization; however, it would be interesting to explore the current approach using character-based tokenization. Taking the limited size of the dataset into consideration, we have used uni-directional LSTM. In case of a relatively bigger dataset, bi-directional LSTM can be used which may lead to even better performances. Exploiting this feedback learning approach on other supervised classification problems could be a thought-provoking idea.

References

[1] Kyunghyun Cho et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *arXiv preprint arXiv:1406.1078* (2014).

[2] *CUDA | GeForce.* URL: https://www.geforce.com/hardware/technology/cuda.

[3] *Faker is a Python package that generates fake data for you.* URL: https://github.com/joke2k/faker.

[4] Wael H Gomaa and Aly A Fahmy. "A survey of text similarity approaches". In: *International Journal of Computer Applications* 68.13 (2013), pp. 13–18.

[5] Maya R Gupta, Nathaniel P Jacobson, and Eric K Garcia. "OCR binarization and image pre-processing for searching historical documents". In: *Pattern Recognition* 40.2 (2007), pp. 389–397.

[6] Mark A Jones and Jason M Eisner. "A probabilistic parser and its applications". In: *AAAI Workshop on Statistically-Based NLP Techniques*. 1992, pp. 20–27.

[7] Gitansh Khirbat. "OCR Post-Processing Text Correction using Simulated Annealing (OPTeCA)". In: *Proceedings of the Australasian Language Technology Association Workshop 2017*. 2017, pp. 119–123.

[8] Guillaume Klein et al. "OpenNMT: Open-Source Toolkit for Neural Machine Translation". In: *Proc. ACL*. 2017. DOI: 10.18653/v1/P17-4012. URL: https://doi.org/10.18653/v1/P17-4012.

[9] Karen Kukich. "Techniques for automatically correcting words in text". In: *Acm Computing Surveys (CSUR)* 24.4 (1992), pp. 377–439.

[10] William B Lund, Douglas J Kennard, and Eric K Ringger. "Combining multiple thresholding binarization values to improve OCR output". In: *Document Recognition and Retrieval XX*. Vol. 8658. International Society for Optics and Photonics. 2013, 86580R.

[11] William B Lund and Eric K Ringger. "Improving optical character recognition through efficient multiple system alignment". In: *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries*. ACM. 2009, pp. 231–240.

[12] William B Lund and Eric K Ringger. "Improving optical character recognition through efficient multiple system alignment". In: *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries*. ACM. 2009, pp. 231–240.

[13] William B Lund, Eric K Ringger, and Daniel D Walker. "How well does multiple OCR error correction generalize?" In: *Document Recognition and Retrieval XXI*. Vol. 9021. International Society for Optics and Photonics. 2014, 90210A.

[14] Kareem Mokhtar, Syed Saqib Bukhari, and Andreas Dengel. "OCR Error Correction: State-of-the-Art vs an NMT-based Approach". In: *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. IEEE. 2018, pp. 429–434.

[15] Lszl Nmeth. *The most popular spellchecking library.* URL: https://github.com/hunspell/hunspell.

[16] *Python-based tools for document analysis and OCR.* URL: https://github.com/tmbdev/ocropy.

[17] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.

[18] Charles C. Tappert, Ching Y. Suen, and Toru Wakahara. "The state of the art in online handwriting recognition". In: *IEEE Transactions on pattern analysis and machine intelligence* 12.8 (1990), pp. 787–808.

[19] *TensorFlow Visualization Toolkit.* URL: https://github.com/tensorflow/tensorboard.