Media Engineering and Technology Faculty

German Research Centre for Artificial Intelligence

Investigating OCR and Post Error Correction Options Using Deep Learning Approaches

Bachelor Thesis

Author:	Omar Morsi
Supervisors:	Prof. Dr. Andreas Dengel
	Prof. Dr. Slim Abdennadher
	Dr.-Ing. Syed Saqib Bukhari
Submission Date:	30 September 2018

Media Engineering and Technology Faculty

German Research Centre for Artificial Intelligence

Investigating OCR and Post Error Correction Options Using Deep Learning Approaches

Bachelor Thesis

Author:	Omar Morsi
Supervisors:	Prof. Dr. Andreas Dengel
	Prof. Dr. Slim Abdennadher
	Dr.-Ing. Syed Saqib Bukhari
Submission Date:	30 September 2018

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor Degree

(ii) due acknowlegement has been made in the text to all other material used

---

Omar Morsi
30 September, 2018

Acknowledgments

After thanking Allah (SWT), I would like to express my gratitude to my mother, father and sister for doing as much as they can to support, help me and motivate me. I would have never reached this far without their support. I would like thank my thesis supervisor Dr.-Ing. Syed Saqib Bukhari for accepting me to work on his project. I would like also to thank him for introducing me to the field of deep learning and optical character recognition (OCR) and helping me getting through a lot of difficulties throughout the whole project. I would like to thank Prof. Dr. Andreas Dengel for accepting me as a part of the team in DFKI. I would like to thank Kareem Mokhtar for the time he spent helping, advising and transferring his knowledge to me.

Abstract

In this work, I investigate and compare different options for post error correction using a neural machine translation (NMT) approach. In this thesis, I describe the different architectures for the optical character recognition (OCR) with and without the neural machine translation (NMT) for error correction. I show the results of the experiments for every architecture along with comparisons between them. I show that in all of the different architectures I examined, the error rate of the optical character recognition (OCR) models combined with the neural machine translation (NMT) models used for error correction is less than the error rate of the optical character recognition (OCR) models without error correction. I also show that the best accuracy **3.814%** for optical character recognition (OCR) combined with neural machine translation (NMT) for error correction is achieved when both models are trained on the whole training dataset. I show that this error percentage is significantly different from the error percentage of **6.313%** of the best trained optical character recognition (OCR) model without error correction.

Contents

Acknowledgments **VII**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Thesis Objective . . . 2
1.3 Thesis Sections . . . 2

2 Literature Review **3**
2.1 Optical Character Recognition (OCR) . . . 3
2.1.1 Applications . . . 3
2.1.2 OCRopus . . . 5
2.2 Machine Translation . . . 6
2.2.1 Introduction . . . 6
2.2.2 Applications . . . 7
2.3 Neural Machine Translation (NMT) . . . 8
2.3.1 OpenNMT . . . 9

3 Error Correction **11**
3.1 NMT for Error Correction . . . 12
3.2 OCR Error Correction Pipeline . . . 14

4 Experimental Methodology **17**
4.1 Targeting Area . . . 17
4.2 Choosing Models . . . 19
4.3 Training Configurations . . . 21
4.4 Data Set Partitioning . . . 23
4.4.1 One-based . . . 24
4.4.2 Half-based . . . 25
4.4.3 One-third-based . . . 26
4.4.4 Two-Thirds-based . . . 27
4.5 Generating Extra OCR Outputs . . . 29

5 Results **31**
5.1 Without Error Correction . . . 31
5.2 One-based Pipeline . . . 32

5.3 One-based Pipeline with multiple OCR outputs 33
5.4 Half-based Pipeline 34
5.5 One-third-based Pipeline 37
5.6 Two-Thirds-based Pipeline 40
5.7 Significance Test 43

6 Conclusion **45**
6.1 Results Summary 45
6.2 Interpretation 46

7 Future Work **47**

Appendix **48**

A Lists **49**
List of Abbreviations 49
List of Figures 51

References **53**

Chapter 1

Introduction

Machine learning has proved its power at solving complex problems at recent years. One of those problems that are being discussed and researched for decades is the optical character recognition (OCR) problem. Teaching a machine how to replicate the function of human eyes and brain is a very hard problem. Teaching a machine how to read is very useful to humanity in a lot of different ways. It will ease and accelerate a lot of daily tasks. Digitizing written or drawn data has been an active research topic recently. It will ease searching in huge documents and books. It will ease editing and finding mistakes. Also, it will facilitate storing important, degraded and torn pages.

1.1 Motivation

In the recent decades machine learning has been improved significantly. Also, the demand for accurate and fast optical character recognition (OCR) systems has been increased. Although nowadays I have optical character recognition (OCR) systems that are able to perform a lot of complex tasks like digitizing scanned documents or even images of original documents, these systems still produce errors while trying to recognize these documents. Since there a lot of sensitive applications for optical character recognition (OCR), it needs to be very accurate. That is why optical character recognition (OCR) error correction has been a very important research topic. Imagine how helpful it is to digitize a written or a printed document with just taking a photo of it with a very small error percentage. Imagine how accurate should license plate recognition be. In this thesis I discuss deep learning approaches for optical character recognition (OCR) error correction. I was motivated about deep learning approaches because of the great power and the great success achieved by it over the last decade in solving very complex problems. For us, the idea of reducing the problem of optical character recognition (OCR) error correction to the problem of translating the optical character recognition (OCR) erroneous output to the correct one was a very motivating and interesting idea to experiment.

1.2 Thesis Objective

The main objective of this bachelor thesis is to show the results of the experiments I performed that proves that optical character recognition (OCR) error correction is a very important process regarding the accuracy of the final output produced. Most of our experiments used the approach discussed in this paper [6]. Also, another goal was to examine whether it is better to divide the data in hand so that some of the data are used to train the optical character recognition (OCR) models and the remaining data are used for error correction neural networks to be trained on or it is better to train both models on the whole training dataset in hand.

1.3 Thesis Sections

The thesis is composed of seven chapters. The second chapter is Literature Review, in which I introduce briefly what is optical character recognition (OCR) and the open source software I used for training models. Also, I present briefly what is neural machine translation (NMT) and the open source software I used for it. In the third chapter, Error Correction, I explain the idea of using neural machine translation (NMT) for optical character recognition (OCR) error correction and I explain how does both optical character recognition (OCR) and neural machine translation (NMT) work together to obtain one pipeline. In the fourth chapter, Experimental Methodology, I talk about the data set I used for training and I show some problems I faced at the beginning with the data set and how I solved those problems.. Also, I talk about the criteria I used to choose the best models to represent the whole pipeline. Also, I talk about the training configurations I used to train the models included in the experiment so as to be easier for future work to proceed quickly with the same configurations or with better ones. At the end of the fourth chapter, I talk about the different partitioning options of the dataset that I experimented. The fifth chapter, Results, I show the outcomes of the experiments I performed for each partitioning. In the sixth chapter, Conclusion, I summarize the results and I give an interpretation for them. In the last chapter, Future Work, I give a recommendation on how can I take this idea to the next level.

Chapter 2

Literature Review

2.1 Optical Character Recognition (OCR)

Optical Character Recognition (OCR) is the electronic translation of handwritten, typewritten or printed text into machine translated images [1]. Optical character recognition (OCR) has been used for decades. Nowadays, OCR is widely used and has many applications. Optical Character Recognition can be used for digitizing handwritten documents, printed documents and also can be used for digitizing historical documents. The task of recognizing handwritten or printed documents is sometimes difficult because of the differences in written styles and the differences in the layouts of the documents, also it can be difficult because of the different backgrounds of the images to be recognized. Nowadays, A lot of researchers around the world are concerned about OCR and inventing new ways of training OCR models. Commonly, documents are preprocessed before used to train OCR or before being digitized by OCR. Preprocessing often includes layout analysis, isolating the characters and binarization of images. The first system using OCR commercially was out in year 1955 at the readers digest. This system used optical character recognition to put the reports of sales to a machine, and after that good use of OCR, it becomes more often to find OCR used for digitizing and computerizing the physical office documents [3].

2.1.1 Applications

Nowadays, there are many applications involving the use of optical character recognition (OCR) as a core phase. One of the applications of optical character recognition (OCR) is license plate recognition which was described in a paper in 2006. An optical character recognition neural network was used in hand with connected components analysis and a new image segmentation technique, which produced a powerful technique for license plate recognition with a performance up to 95%, which unfortunately can still cause a lot of problems [7]. Another interesting application is scanning mail information using optical

character recognition (OCR) techniques. One can automate the process of processing, recognizing and delivering mails by recognizing destination countries, cities, states and zip codes written on emails [5].

Optical character recognition (OCR) can also be used for recognizing highly sensitive and important documents like passports and bank statements. There are automated document recognition methods which uses documents templates to extract information from the input image of a user. The image of the user's document is being analyzed with the use of optical character recognition based on the given template documents to spot the places of the text zones.

2.1.2 OCRopus

OCRopus is a free open source optical character recognition (OCR) project. OCRopus was developed in the German Research Centre for Artificial Intelligence in Kaiserslautern, Germany with the lead of prof. Thomas Breuel. The project was sponsored by Google. It was announced as a Google-sponsored open source OCR system project 2007. The project's main objective was to deliver a reliable high-quality optical character recognition system that can be used for historical document analysis, electronic libraries and document conversions [2]. OCRopus is the open source software that I used in the experimentation process for training, validating and testing our OCR networks. Before applying OCRopus on documents, one may need to do some image preprocessing so as to get better results. AnyOCR can handle important image preprocessing operations like image binarization, layout analysis, text line recognition & cropping. Also. You can find the workflow diagram of the separate command line tools from OCRopus in figure 2.1.

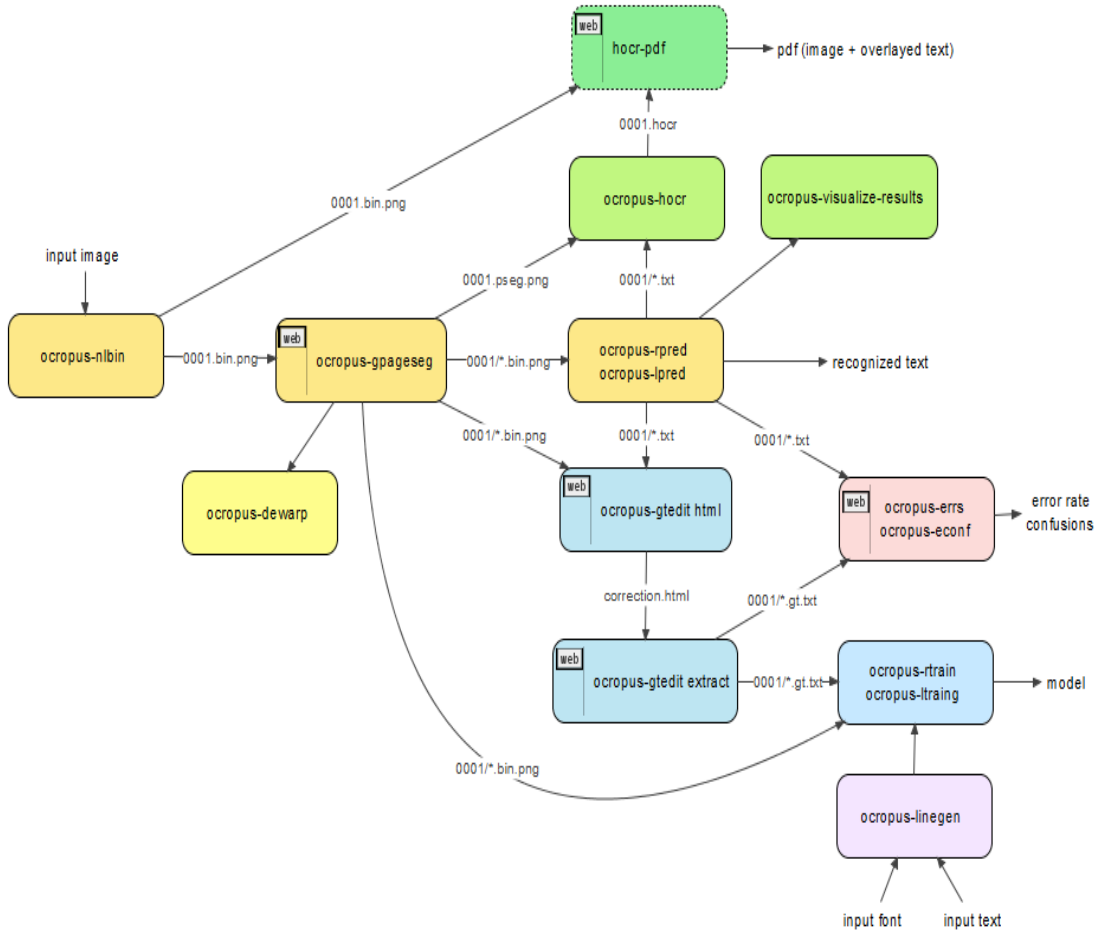

Figure 2.1: Ocropus Workflow of the separate command line tools from OCRopus. ocropus-econf, ocropus-rtrain ocropus-rpred are the 3 commands used through our experiments.

2.2 Machine Translation

2.2.1 Introduction

Machine translation is converting a source text that I can represent as a sequence of characters from a certain language into another text in another language using machines. Machine translation is sometimes referred to as MT. Yehosha Bar-Hillel was the first one who researched in this field and he started researching in MIT. There are approaches for machine translation that relies on linguistic rules and dictionaries for languages as the Rule-Based Machine Translation (RBMT) approach. There are three methods for rule based machine translation (RBMT) which are the direct method, transfer-based method and interlingua method as in figure 2.2 [12]

In the direct method, the words in the input or the source language is directly trans-

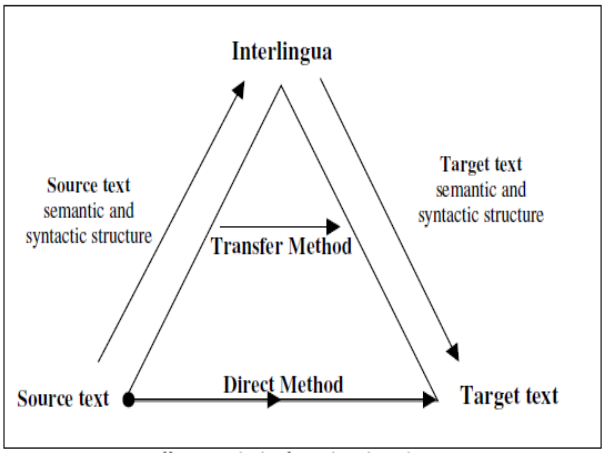

Figure 2.2: Rule Based Machine Translation Methods.

lated without being passed by any other representations. In the interlingua method the input language is transformed into some kind of an intermediate language and after that, the target language will be derived from the intermediate language. In the transfer based method, the input language is transformed into some kind of a less language-specific representation of which the target language is deprived [12].
Another approach for machine translation is the statistical machine translation approach (SMT). This approach deeply depends on rules obtained from language experts and text corpora which are usually probabilistic. This approach is not easy to apply due to the unfamiliarity of linguistics researchers to the mathematics involved. For statistical machine translation any string of words e is a candidate translation for a sequence of words f. The goal is setting a probability distribution $P(e|f)$ over all the pairs of strings of the two languages. After that, given any string f, one can find the string e that maximizes the probability $P(e|f)$ [4].

Another approach to machine translation is the neural machine translation (NMT) approach which will be described in the next section 2.3.

2.2.2 Applications

Machine translation has many applications and usages. The most intuitive application is translating sentences from a source language to a target language. But machine translation can be used for far more than just translating text from one language to the other. It can be used for translating speech in a language to text as it was applied in Moses which is one of the open source toolkits for machine translation [11]. Also, machine translation can be used for speech to speech translation, which is converting one person's utterance from one language to another language as it was pointed out by the inventor Timothy Alan Dietz as he described a method for speech recognition with text-based language conversion and text-to-speech in a client-server configuration. Machine translation can

be used for object recognition as well, in which translation from image of words to actual words in a certain language is produced.

2.3 Neural Machine Translation (NMT)

Neural machine translation (NMT) is a newly introduced approach to machine translation. It is a relatively new approach if it is compared to statistical machine translation (SMT). Unlike statistical machine translation which requires having statistical models which have their parameters obtained by analyzing large bilingual text sets, the neural machine translation approach depends purely on neural networks. Among all the discussed neural machine translation models the common structure includes having two recurrent neural networks one is called an encoder and the other one is called a decoder. The neural network of the encoder takes as an input the source sentence and encodes it into a fixed-length vector. The decoder is responsible for outputting a translation for the input sequence of words given the encoded vector as shown in figure 2.3. The whole system consisting of the encoder and the decoder should together be trained to maximize the chance of getting a correct translation when given a source sentence as an input [9].

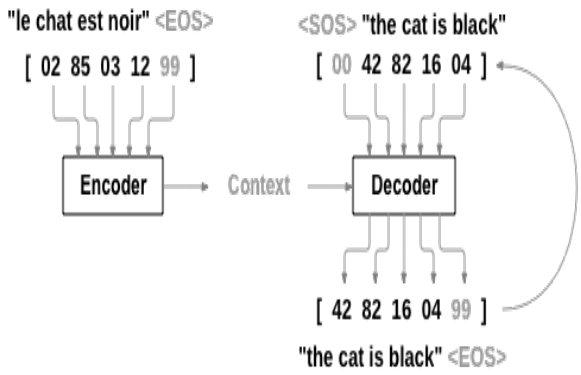

Figure 2.3: Encoder-Decoder Neural Network Architecture.
[1]

One of the Encoder-Decoder frameworks called RNN Encoder-Decoder was proposed in 2014. In this framework the encoder gets a sequence of words $x = (x_1, ..., x_n)$ and encodes it into a vector c such that

$$c = q(h_1, ..., x_n)$$

and

$$h_t = f(x_t, h_{t-1})$$

[1] https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html

here h_t is a hidden state of time t. After that, the decoder's goal is to predict the next word given the vector c and all the words that were predicted before it $y_1, ..., y_{t-1}$. This is done by training the neural network to get the word y_t which will maximize the probability of $p(y) = p(y_t|y_1, ..., y_{t-1}, c)$ [9].

2.3.1 OpenNMT

OpenNMT is an open source code used for neural sequence modeling and for neural machine translation. OpenNMT was launched in December 2016. OpenNMT has three main implementations [8]:

- OpenNMT-lua: this implementation is the original one. OpenNMT-lua was developed with LuaTorch. It supports all the features and it is a stable and optimized code which is ready for use to perform experiments.

- OpenNMT-py: this implementation is a clone of OpenNMT-lua but it uses the more modern PyTorch. This implementation was created by Facebook AI research team. It was used to produce models used for neural machine translation. It is a nice implementation because PyTorch is an optimized tensor library which supports tensor computation but with the use of GPUs, which makes the computation accelerated.

- OpenNMT-tf: This is the most recent implementation. It is based on TensorFlow. This approach is effective in large experiments and the experiments that need very high performance. Also, TensorFlow is very powerful as it offers APIs for experts to develop for mobiles, desktops, and cloud. TensorFlow is an open source library that anyone can use for production or for just researching.

I used OpenNMT-py during performing our experiments. Mainly it was used to build the neural network architectures used for neural machine translation in the error correction part.

Chapter 3

Error Correction

The problem of optical character recognition (OCR) error correction is a great challenge, as the goal is to correct errors produced from efficiently trained optical character recognition models. There were a lot of approaches trying to tackle this problem. Researching this problem is very important as sometimes mistakes obtained from optical character recognition (OCR) models are crucial due to the sensitivity of the applications for which optical character recognition (OCR) is used.

Errors in a sentence are errors produced by one of the following two categories. The first type is real word errors. Which means errors that were not intended to be made by the source writer of the document. This type of error is very hard to detect and need much knowledge to tackle. Non-word errors are errors produced by the optical character recognition (OCR) system itself. For example, if the source text "I love my car" was recognized by the optical character recognition network as "I love my cat", then the letter "t" in "cat" is a non-word error. In our investigation, I focus on the non-real word errors only, which means that I assume that the source documents provided contains valid words and need not to be corrected. Mainly, I train optical character recognition models and I focus on correcting errors produced by the wrong recognition of the OCR models.

Optical character recognition (OCR) error is an important stage because although optical character recognition (OCR) technology nowadays is very powerful, they have limitations. Factors like poor quality documents, old documents containing torn faces and ancient scripts restricts how good and effective the optical character recognition alone can be. That is why optical character recognition error correction is important. With a good optical character recognition error correction network one can increase the accuracy produced by the whole pipeline if the OCR error correction model was trained well to have the power to recognize the patterns where the OCR is most likely to fail at. This can be imagined as the OCR error correction network is trained to know when it is most likely in the document or in a sequence of characters written by the source of the document to be recognized incorrectly by the optical character recognition model and what is the usual correct output that should have been produced. For example, an OCR error correction model trained that the OCR usually fails when the character 'c' comes

after 'o' should correct the outputs of OCR when it finds such patterns produced by the OCR. The optical character recognition (OCR) network can produce errors due to several reasons. For example, one of these reasons is that characters are linked together due to the nature of the written document or due to the way that the writer of the document is used to write. This linkage makes it difficult for the optical character recognition (OCR) network to detect the intended characters because maybe after the linkage of a certain character, it had extra connected parts linked with it which can make it look like another character. From our point of view, a well trained error correction network can detect the sequence of characters which can confuse the OCR network and can correct it as well.

3.1 NMT for Error Correction

As it was suggested by the following paper [6], I used neural machine translation (NMT) for post-OCR error correction to improve the accuracy of the OCR system. I used OpenNMT to build the neural networks for the error correction stage. In the paper, there were two architectures suggested for the error correction. The main idea was to deal with the correction problem as if it was a translation problem. The translation problem basically deals with two languages. One language is the source language and the other one is the target language. And given statements in the source language, the goal is to translate it into the target language. Projecting this on the optical character recognition problem was a brilliant idea. I deal with the OCR output, which might contain a lot of mistakes, as the source language. Also, I deal with the ground truth that I have in hand for the incorrect OCR output as the target language that I want to translate the OCR output into. Here comes the role of the neural machine translation (NMT) network which will take the OCR output as the source language and the corresponding ground truth as the target language and trained to translate between them. The two models introduced in the paper were:

- Word Based Sequence-to-Sequence model This model basically works on a word level of the source language or the output of the OCR. The words in the dictionary of the source and the target languages are extracted from the OCR output and the ground truth respectively. Before the training process starts one should build the dictionaries of the source language and the target language and should also tokenize the sentences so that every word is separated from all other words and from the punctuation as well as shown in figure 3.1

  One of the huge disadvantages of this model is that it needs a huge amount of parallel data to be trained on, so as to learn the relation between sequences of words. But one of the advantages is that in the word-based model when trained, the neural network will learn the relation between words that construct a correct sentence.

- Character-Based Sequence-to-Sequence model: This model was the second suggested model. The difference between this model and the word-based sequence

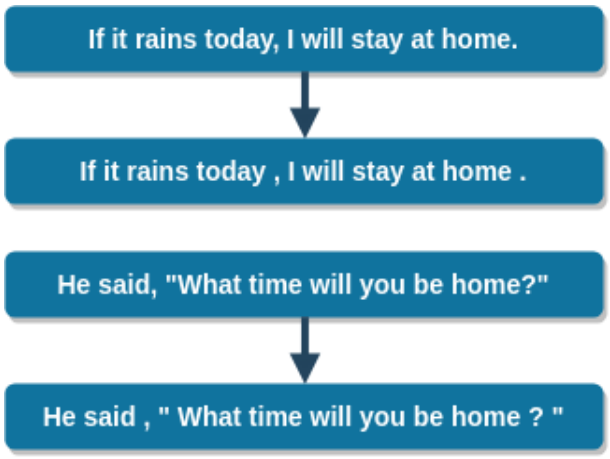

Figure 3.1: Word Based Sequence to Sequence model examples illustrating how are the words should be separated from each other as well as from other punctuation characters.

to sequence model is that the dictionary of the source language is built using the characters of the OCR output, not the words as it was the case in the word-based model. Also, the dictionary of the target language is built on the characters of the ground truth. The main advantage here is that now I don't need the same amount of data to train the neural network on as I need in the first model. This model is the one I used to perform our experiments as it showed better results in the paper when used on the Latin dataset that I are using for training as shown in figures 3.1 and 5.5 [6].

Dataset	**Before Correction**		**After Correction**		**Relative Improvement**	
	CER	WER	CER	WER	CER	WER
English	2.87%	8.87%	8.42%	10.10%	-5.71%	-1.35%
German	10.65%	47.12%	44.69%	39.15%	-38.11%	15.07%
Latin	8.71%	33.11%	182.94%	109.70%	-190.86%	-114.49%

Table 3.1: Word Based Sequence to Sequence model accuracy on English, German and Latin data sets.

Dataset	**Before Correction**		**After Correction**		**Relative Improvement**	
	CER	WER	CER	WER	CER	WER
English	2.87%	8.87%	3.33%	8.82%	-0.47%	0.06%
German	10.65%	47.12%	15.12%	34.01%	-5.01%	24.78%
Latin	8.71%	33.11%	7.63%	22.34%	1.18%	16.10%

Table 3.2: Character Based Sequence to Sequence model accuracy on English, German and Latin data sets.

3.2 OCR Error Correction Pipeline

In this section, I will discuss the pipeline of optical character recognition along with error correction. Our input is the corpus of images for statements along with their ground truth written in text documents.

- The first stage is training the OCR model using the corpus of the images along with the ground truth as an input. As shown in figure 3.2 the OCR inputs are *images1* and *ground_truth1*. *images1* is the set of images used for training the OCR model as it may be completely different than the set of images used for training the NMT model, also it can overlap with the set of images used for training the NMT model as I will show in chapter 4. *ground_truth1* represents the set of documents containing the ground truth corresponding to images in *images1*.

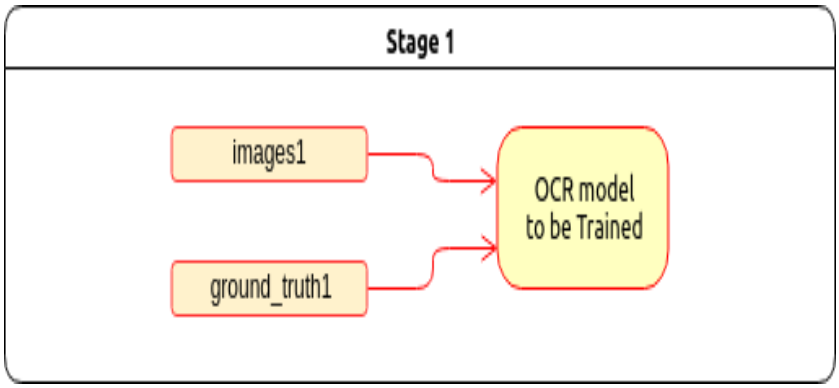

Figure 3.2: Stage one of OCR Error Correction showing how does the OCR models use the training dataset specified for OCR to be trained on .

- The second stage is to train the neural machine translation (NMT) model that will be used in error correction. For this stage, I use the best trained OCR model to predict input images that I have their ground truth as well. The output from the OCR prediction as an input along with the ground truth to the neural machine translation (NMT) model that I want to train as shown in figure 3.3.

  In the second stage, the only input for the trained OCR model is *images2*, which is the set of images that will be used to be predicted by the OCR. Now I have the optical character recognition (OCR) model along with the neural machine translation (NMT) model. *ground_truth2* represents the set of documents containing the ground truth corresponding to images in *images2*.

In 3.4, I show how to predict an image using the trained optical character recognition (OCR) model along with the trained neural machine translation (NMT) model. To

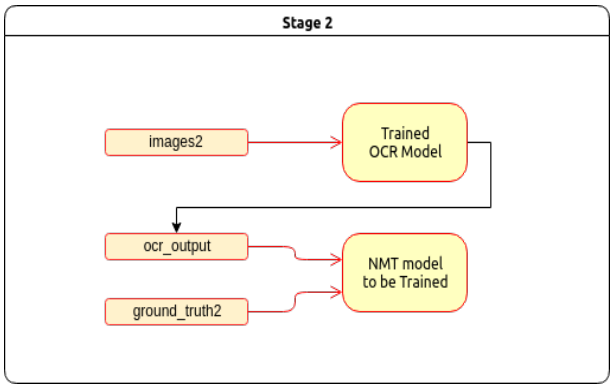

Figure 3.3: Second stage of OCR Error Correction, showing how does the neural machine translation (NMT) models train using the best trained optical character recognition (OCR) model's prediction on the training dataset specified for the neural machine translation (NMT) network.

predict a test input image I provide it to the OCR model I trained as an input. The prediction of the OCR model is provided to the trained NMT model as an input. The output from the trained NMT model is the final output that I claim is better than the OCR output as I will show the results.

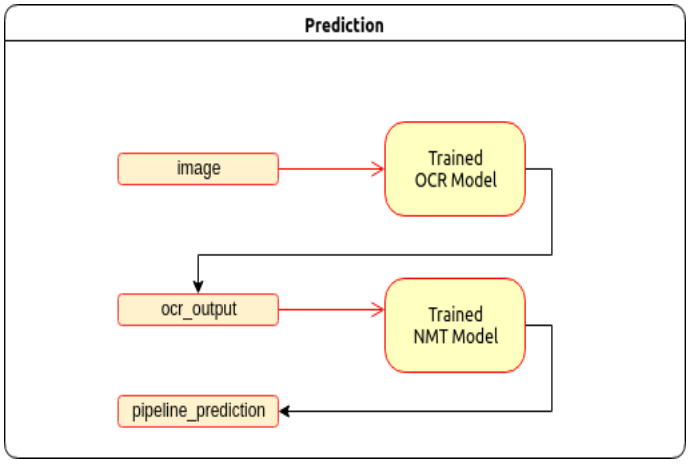

Figure 3.4: OCR Error Correction Predicton Steps showing how does the best trained optical character recognition (OCR) model is used with the best trained neural machine translation (NMT) model for predicting input text.

Chapter 4

Experimental Methodology

4.1 Targeting Area

In this section, I will talk about the targeting area that I focused on for training, validation and testing. I focused on historical documents. That is why I used the dataset that I worked on for performing the experiments is a Latin dataset taken from a Latin book from the 15^{th} century. The book is an old one written using an old font as shown in figures 4.1 and 4.2.

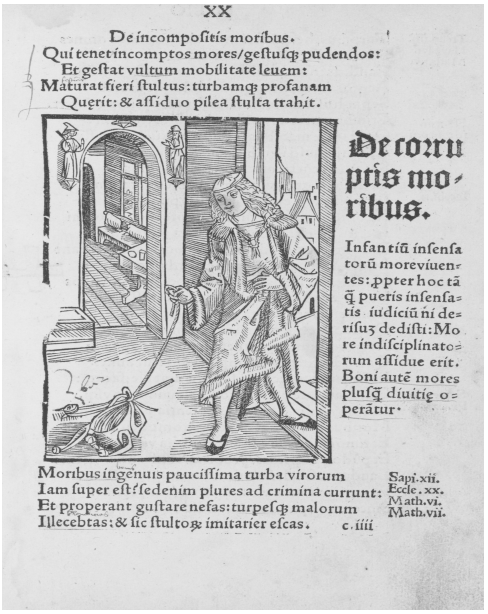

Figure 4.1: Latin Book Page.

I didn't work on the pages of the book directly. I worked on extracted data from the book using anyOCR for segemntation. AnyOCR is an end-to-end framework that combines segmentation-free OCR methods with segmentation-based methods using LSTM

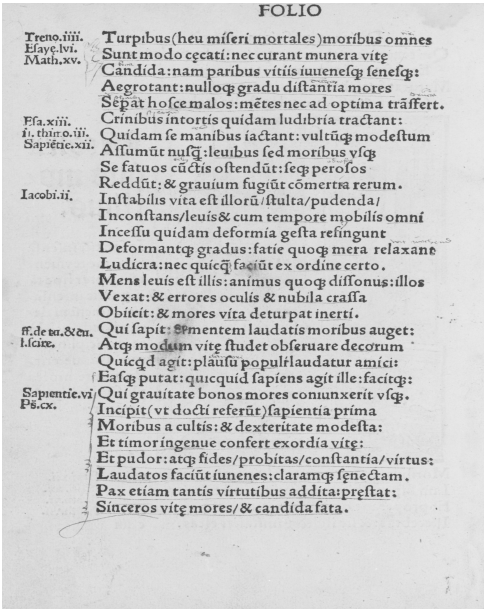

Figure 4.2: Latin Book Page.

architectures which proved high accuracy with just 8% error on the segmentation of the dataset [10].
I worked on latin dataset after it was segmented and binarized into 3418 different parts. Every part has its own segmented and binarized image in a PNG format along with its ground truth in a TXT format as shown in figure 4.3 two lines segmented from the book with their corresponding digitized ground truth.

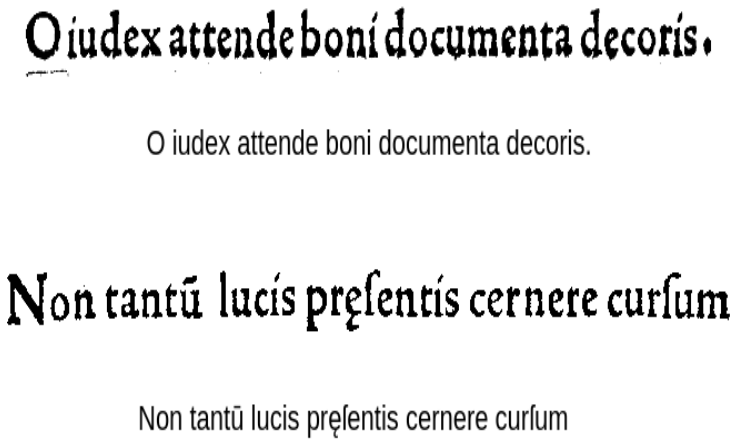

Figure 4.3: A sample of the dataset I are working on. Showing two Cropped and Binarized Parts from the Latin Book along with their corresponding digitized ground truth.

There was some problems due to segmentation in the files provided which would have made problems regarding the accuracies of both the OCR models and the NMT models I trained, like having some ground truth files with tabs at the beginning of the sentences or some tabs at the end of the sentences. as shown in 4.4 and 4.5.

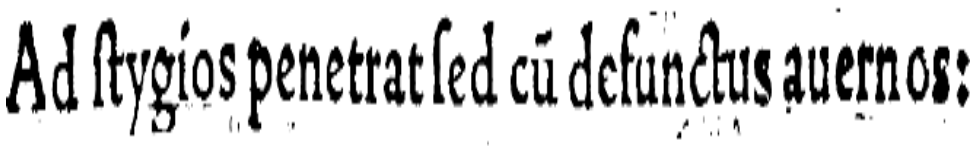

Figure 4.4: Sample From the Latin Dataset images.

Figure 4.5: Problem with the latin dataset was having tabs before the beginning of the sentences in the ground truth files.

4.2 Choosing Models

For every architecture I worked on, I trained the models of the OCR and the NMT for at least 30 epochs. To choose the right model to represent the OCR in any of the architectures I relied on its performance on the validation dataset. The validation dataset is 10% of the whole dataset I have. The OCR models do not train on any line from the validation dataset so as the choosing of the right OCR model is not biased. Measuring the error rate of an OCR model on the validation dataset is done using computing the edit distances between ground truth and recognizer output. I stop training the OCR model if one of the following occurred:

- The last 5 trained models have error rates on the validation dataset higher than the model with the best error rate on the validation dataset.
- The difference between the least error rate and the best error rate of the five models with the best error rates trained so far less than 0.1

After I stop training I choose the OCR model with the least error rate on the validation dataset to be the model that I will continue working with.

Epochs	Error%
54	7.479
55	6.487
55	8.482
56	6.754
57	6.609
58	6.899
58	6.855
59	6.375
60	6.609
61	6.576
61	6.375
62	6.197
63	6.721
64	7.98
64	7.3
65	6.353
66	7.323

Table 4.1: Error Rates of optical character recognition (OCR) models on validation dataset.

As shown in table 4.1, I stopped training because the second condition is satisfied and I have chosen the OCR model trained on 62 epochs because it has the best error rate over all the models trained before stopping.

I do the same with the neural machine translation models that I train to figure out the best NMT model to choose. The validation dataset I use to choose the best NMT model is always the same as the validation dataset I used to choose the OCR model of which the output is the input used to train the NMT models. The difference is that for the NMT models to be trained, OpenNMT requires just two input files one for the sentences of the source language separated by a line break and the other for the sentences of the target language separated by a line break. Also, for the NMT model to translate, OpenNMT requires just two input files. So to measure the validation error rate for an NMT model, I implemented a script *validate-nmt.py* to do this process. This script takes 4 inputs:

- The NMT model I are testing.
- A folder containing:
  - The validation dataset images in .png extensions.
  - The validation dataset ground truth in .gt.txt extensions.
  - The OCR output files in .txt extensions.

For every image in the validation dataset there should be a corresponding ground truth and an ocr output files with the same file names.

In the script, I loop over the files of the OCR output one by one doing the following:

- Tokenizing the OCR output file to match the character-based model criteria mentioned here [6].
- Using the NMT model to translate the tokenized file.
- Detokenizing the output file.

After I are done with all the files I use the script *ocropus-econf.py* provided with the ocropus implemntation to measure the edit distance and calculate error rate using the same method I did for the selection of the OCR model so as the results are not biased.

4.3 Training Configurations

In this section, I will show the configurations and the steps I followed to train optical character recognition models as well as neural machine translation models. As I previously mentioned, I use OCRopus for training OCR models. One of the difficulties that one meet while working on a latin dataset, is that it contains characters that are not encoded in ASCII. By default, when training an OCR model, the implementation that worked with [1]includes some characters to be recognized as shown in figure 4.6. Those characters are not enough to recognize all the characters that I want for the latin data. Thats why it was important to construct a codec from the actual ground truth dataset so as to include as much characters from the latin dataset as I can as shown in figure 4.7.

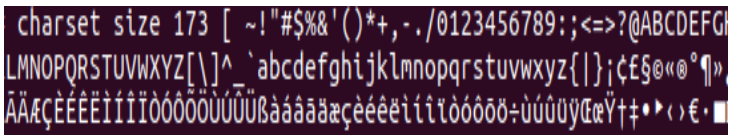

Figure 4.6: Old Set of Characters which is the OCRopus default character set.

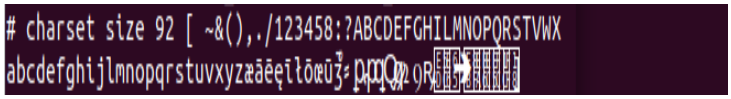

Figure 4.7: New Set of Characters obtained by specifying to OCRopus to construct a codec from the training ground truth files.

This increased the efficiency of the OCR models trained according to removing unneeded characters and inserting needed characters that were not there in the default

[1]https://github.com/tmbdev/ocropy

codec.
Following are the rest of the configurations needed to train the OCR models to obtain the same accuracy I reached.

- LSTM learning rate: 0.0001
- LSTM state units: 100

After that ocropy requires mentioning the path for the .PNG files on which the models will be trained on. Also, it requires having the ground truth files in the same directory with the same names as their corresponding .PNG file but with an extension .gt.txt. That is all needed to train the OCR models to obtain the accuracies I will mention in chapter 5.

As mentioned previously, for training NMT models, I used OpenNMT. Before training the OpenNMT model, I must train the OCR models, select the best one of them according to the validation dataset and to let the best model predict the output on the training dataset of the NMT model to use its output as an input for training the NMT models. OpenNMT requires the source text to be in one file as statements separated by line breaks. For collecting all of the OCR output text files into one file I implemented `collect.py` which takes as an input : - Source Directory: The directory where the files to be collected reside. - Target File: The file in which the text in the files will be written. - Extension: The extension of the files to be collected. For collecting OCR output files I need to pass the directory where the files to be collected reside with an extension of .txt, because this is the default format of the predicted output of the OCR. For collecting the ground truth files I need to pass the directory as well as an extension of the ground truth files which in our case is .gt.txt. The script sorts the files obtained from this directory with this extension so as to put the statements in order to match the statements in the other file.
After collecting the text from the files I tokenize both files by letters to match the character-based-model criteria mentioned here [6].
Finally, I use the following configurations to train the NMT model on the training dataset that I prepared using OpenNMT.

- Number of Layers: 4
- Batch Size: 64
- Learning Rate: 0.001
- RNN Size 1024
- Optimizer: adam
- Input Feed: 3
- Dropout: 0.3

4.4 Data Set Partitioning

In this section, I will talk about the different ways of partitioning the dataset that I experimented. Mainly the ratio of the training dataset to the validation dataset to the test dataset is fixed in all of the experiments with 80% for the training dataset, 10% for the validation dataset and 10%. What I mean by partitioning is mainly partitioning the training dataset into two parts, one of them is the one on which the optical character recognition model will train, and the other part is the one on which the OCR model will predict and the output will be the input of the NMT. So the other part is mainly the part used for training the NMT. I also tried other variants as I will show in the next subsections.

The validation dataset is always 10% of the whole dataset in all of our experiments. It is used to choose the best model for both of the optical character recognition model as well as the neural machine translation model.

The test dataset is always 10% of the whole dataset in all of our experiments and it is used to test the error rate of the optical character recognition model as well as the whole pipeline -OCR + NMT-. The test dataset is never included in the training process so as not to bias the results.

In the four following subsections, I explain how I partitioned the dataset. $\boldsymbol{x}$-based partitioning refers mainly to the fraction of the training dataset I used to train the OCR model. After that in each part I experiment the efficiency of both OCR model alone and the OCR model when combined with NMT in two different ways:

- Training the NMT on 1 - $\boldsymbol{x}$ of the training dataset. Which means using the 1 - x fraction of the training dataset remaining as inputs to the OCR after it is trained and the predicted output is used afterward as an input to the NMT model going to train.
- Training the NMT on the whole training dataset. Which means using all of the training datasets as inputs to the OCR after it is trained and the predicted output is used afterward as an input to the NMT model going to train.

So as to explain the following subsections easier, I will use some notations:

- trainI-1-1, trainG-1-1 : The images and ground truth documents of the whole training dataset respectively.
- trainI-1-2-a, trainG-1-2-a: The images and ground truth documents of the first half of the training dataset.
- trainI-1-2-b, trainG-1-2-b: The images and ground truth documents of the second half of the training dataset.

- trainI-1-3-a, trainG-1-3-a: The images and ground truth documents of the first third of the training dataset.
- trainI-1-3-b, trainG-1-3-b: The images and ground truth documents of the second third of the training dataset.
- trainI-2-3-a, trainG-2-3-a: The images and ground truth documents of the first two-thirds of the training dataset.
- trainI-2-3-b, trainG-2-3-b: The images and ground truth documents of the second two-thirds of the training dataset.

For all of the next architectures mentioned, any input to train the NMT model is tokenized using the character-based sequence-to-sequence model explained here [6].

4.4.1 One-based

In One-based partitioning, I train the OCR models on 100% of the training dataset I have. Afterward, I train the NMT models on the whole training dataset as show in figure 4.8.

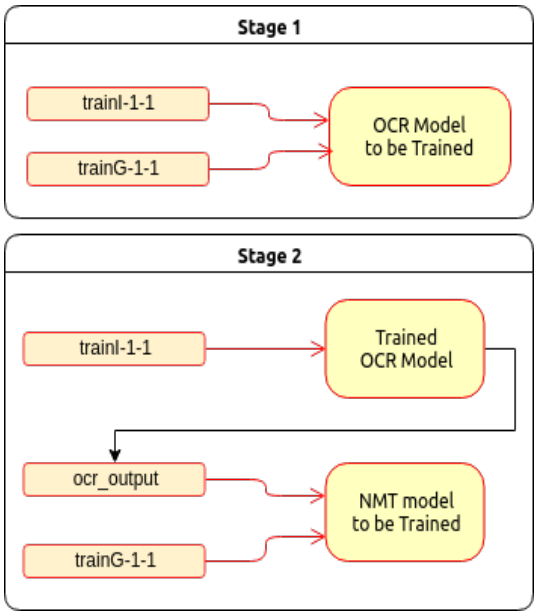

Figure 4.8: Showing the two stages needed for training optical character recognition (OCR) models and neural machine translation (NMT) models based on the one-based partitioning. The optical character recognition (OCR) and the neural machine translation (NMT) models are both trained on the whole training dataset.

4.4.2 Half-based

In Half-based models, I try two approaches. The first approach is to train the OCR models on the first half of the training dataset and to train the NMT models on the whole training dataset using the best trained OCR model prediction on the whole training dataset as an input along with the ground truth as shown in Figure 4.9. The second approach is to train the OCR models on the first half of the training dataset and to train the NMT models on the second half of the training dataset as shown in figure 4.10. The second approach was promising as the NMT uses the prediction of the best OCR model trained on a dataset that it wasn't trained on initially.

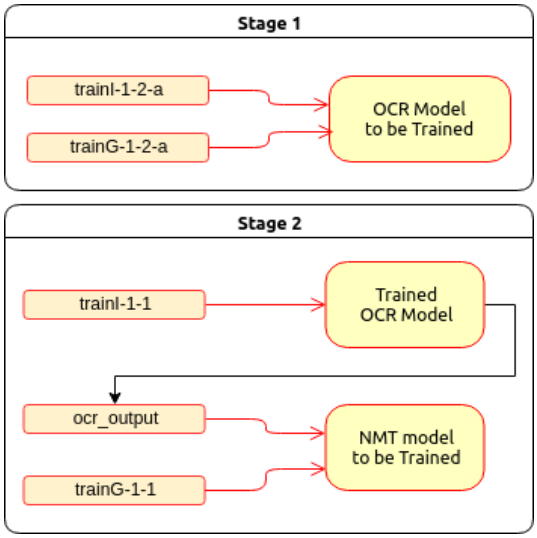

Figure 4.9: Half-based partitioning Diagram showing stages ones and two when the neural machine translation (NMT) models are trained using the prediction of the optical character recognition (OCR) model on the whole training dataset.

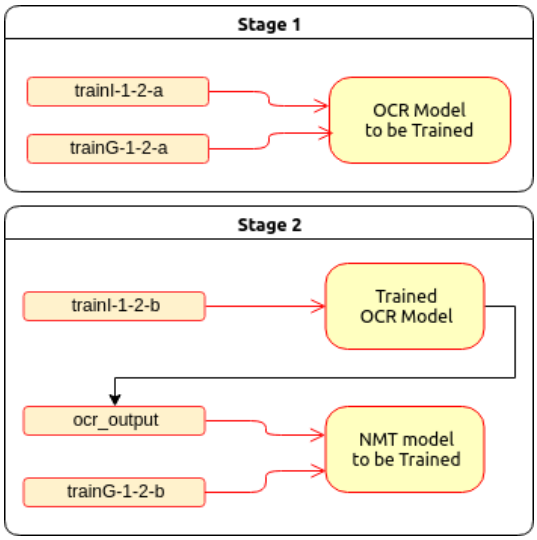

Figure 4.10: Half-based partitioning Diagram showing stages ones and two when the neural machine translation (NMT) models are trained using the prediction of the optical character recognition (OCR) model on the second half of the training dataset.

4.4.3 One-third-based

In the One-third-based approach for dataset partitioning, I do the same I apply our two approaches but with a fraction of **1/3**. In this approach, the OCR model is trained on a dataset which is smaller than the one the NMT is trained on. The OCR model is given the first third of the training dataset to be trained on. Afterward, the NMT model is trained using the best OCR model trained prediction for the whole training dataset as an input along with the ground truth one time as shown in figure 4.11 and using the best OCR model trained prediction on the second two-thirds of the training dataset as an input along with their corresponding ground truth as shown in figure 4.12.

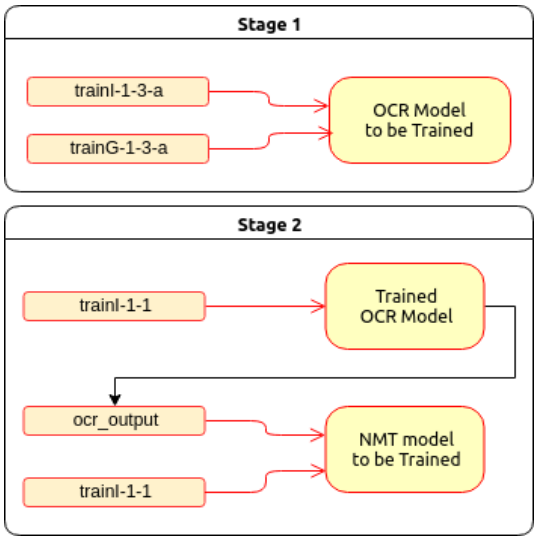

Figure 4.11: One-third-based partitioning Diagram showing stages ones and two when the neural machine translation (NMT) models are trained using the prediction of the optical character recognition (OCR) model on the whole training dataset.

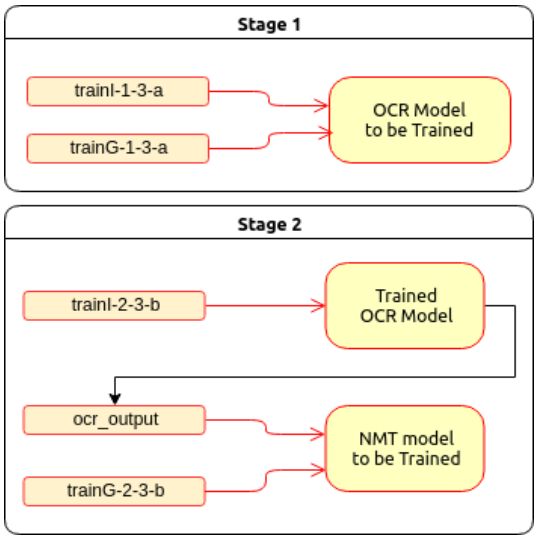

Figure 4.12: One-third-based partitioning Diagram showing stages ones and two when the neural machine translation (NMT) models are trained using the prediction of the optical character recognition (OCR) model on the second two thirds of the training dataset.

4.4.4 Two-Thirds-based

In our last approach of partitioning the dataset, I experimented with partitioning it with a 2 / 3 portion of the training dataset for the OCR models to be trained on. As shown

in figure 4.13, the OCR models are trained on the first two-thirds of the training dataset, and the prediction of the best OCR model trained on the whole training dataset is used as an input along with the corresponding ground truth for the NMT models to be trained on. The second way I experimented for this partitioning is by using the prediction of the best OCR model trained on two-thirds of the training dataset for the second one-third of the dataset as an input for the NMT models to be trained on as shown in figure 4.14.

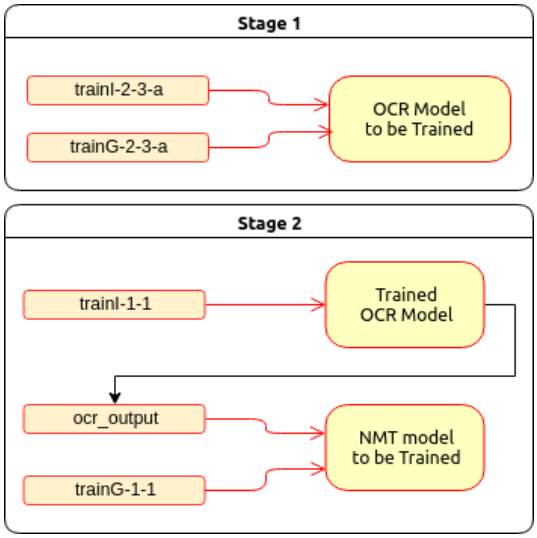

Figure 4.13: Two-thirds-based partitioning Diagram showing stages ones and two when the neural machine translation (NMT) models are trained using the prediction of the optical character recognition (OCR) model on the whole training dataset.

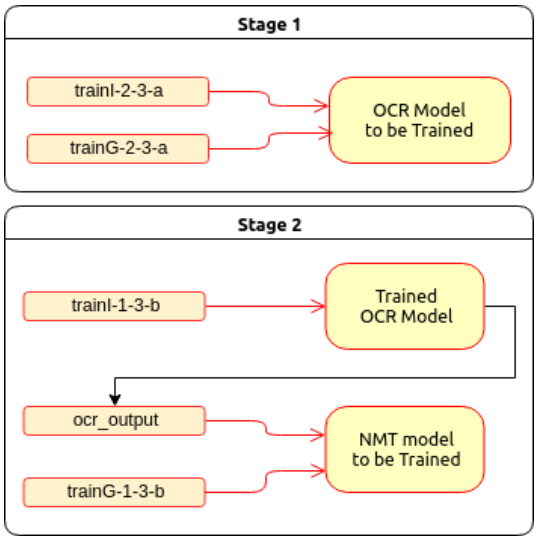

Figure 4.14: Two-thirds-based partitioning Diagram showing stages ones and two when the neural machine translation (NMT) models are trained using the prediction of the optical character recognition (OCR) model on the second one third of the training dataset.

4.5 Generating Extra OCR Outputs

In this section, I explain another approach I examined in an attempt to increase the accuracy of the pipeline. For this approach, I use the same hierarchy as for the One-based partitioning except for the input to the neural machine translation (NMT) model to be trained. In this approach, I use the data in the softmax layer of the optical character recognition (OCR) model's LSTM to capture the characters with which the optical character recognition (OCR) model is confused. For every position in the OCR output, I get all the characters which had a probability more than or equal to 0.3 as the predicted character. And using a brute force algorithm *generate_lines.py* as shown below, I generate all possible combinations of lines made by these characters. For every line, I pair it with the corresponding ground truth line and feed that to the neural machine translation (NMT) model to be trained. In the code shown below, there are two inputs to the search function **maxima** and **outputs**. **maxima** is a set of ordered pairs (**r**, **c**). **r** represents one of the positions in the image which is most probably a character and **c** is the character with the highest probability to be in this position. **outputs** is a 2D array. Rows in this 2D array corresponds to positions in the image, columns corresponds to characters from the constructed codec from the ground truth and cells in the 2D array contains the probability of the corresponding character being the in the corresponding position in the image.

```
candidates = []
res = []
arr = []

def brute_force (idx) :
    if idx == len(candidates):
        copy = []
        for x in arr:
            copy.append(x)
        res.append(copy)
        return
    else :
     for x in candidates[idx]:
            arr[idx] = x
            brute_force(idx + 1)

def search (maxima, outputs) :
    global arr
    global candidates
    for (r, c) in maxima:
        best = []
        max = 0
        for i in range (0, len(outputs[r])):
            if outputs[r][i] > outputs[r][max]:
                max = i;
            if(outputs[r][i] >= 1.0 / 3):
                best.append(i)
        if len(best) == 0:
            best.append(max)
        candidates.append(best)
    arr = [None] * len(candidates)
    brute_force (0)
    return res
```

Chapter 5

Results

In this section, I show the results of the different partitionings I mentioned in the previous chapter. For every partitioning, I will show the error rate of the optical character correction (OCR) models on the validation dataset and I will show which model I have chosen to represent the OCR models. Afterward, I show the error rate of the chosen model on the part of the training dataset on which it trained to show the training accuracy. Also, for every partitioning, I show the error rate of the OCR model on the test data before I use it for training the NMT. I also show the error rate of the NMT models I trained on the validation dataset and I show which model I have chosen to represent the NMT in stage 2. Afterward, I show the error rate of the NMT model I have chosen on the training dataset on which it was trained. Finally, I show the error rate of the chosen NMT model on the test dataset, which shows the accuracy of the whole pipeline.

5.1 Without Error Correction

In this section, I show the data related to the training only the OCR model to obtain the best accuracy on the training dataset without any post error correction process involved. I show the error rate of the OCR models I trained on the whole training dataset on the validation dataset from epoch number 28 till epoch number 34 for models trained on 75000 lines till 91000 in table 5.1.

Lines	Epochs	Error%
75000	28	6.074
76000	28	5.974
77000	29	6.019
78000	29	6.152
79000	30	6.097
80000	30	6.23
81000	30	6.264
82000	31	6.197
83000	31	6.03
84000	31	6.152
85000	32	5.974
86000	32	6.074
87000	33	5.985
88000	33	6.175
89000	33	6.175
90000	34	6.086
91000	34	6.297

Table 5.1: Error Rates of trained OCR models on the validation dataset along with the number of epochs for which every model is trained.

Here the best candidate model is the model with accuracy **5.974%** on the validation dataset.
This particular model got an error rate of **5.222%** on the training dataset. Also, it achieved an error rate of **6.313%** on the test dataset as show in table 5.3.

Dataset	Error%
Validation Dataset	5.974
Training Dataset	5.222
Testing Dataset	6.313

Table 5.2: Error Rates of the best trained OCR model on the validation, training testing datasets.

5.2 One-based Pipeline

In this section, I will show the results of the optical character recognition (OCR) with neural machine translation (NMT) pipeline when the OCR models are trained on the whole training dataset and the NMT models are trained on the whole training dataset as well. I show the error rate of the last NMT models trained on the validation dataset

in the table 5.3. I picked the model trained for **48 epochs** as the representative for the NMT models as it achieved an error rate of **4.7%** on the validation dataset. I also show the error rate of the whole pipeline on the validation dataset and the testing dataset in table 5.4.

Epochs	Error%
48	4.7
49	4.8
50	4.8
51	4.8
52	4.9
53	4.9
54	4.8
55	4.9

Table 5.3: Error Rates of last trained NMT models on the validation dataset along with the number of epochs on which each model is trained.

Dataset	Error%
Validation Dataset	4.7
Testing Dataset	3.814

Table 5.4: Error Rates of the best trained NMT model on the validation testing datasets.

5.3 One-based Pipeline with multiple OCR outputs

In this section, I show the results of the pipeline consisting of an optical character recognition (OCR) model which is trained on the whole training dataset combined with a neural machine translation (NMT) model trained on the whole training dataset after generating multiple OCR outputs for every image with the use of the OCR model's LSTM's softmax layer. First, I show the error rates of the last neural machine translation (NMT) models trained with respect to the validation dataset in table 5.5. I also show the validation error rate with the testing error rate of the best neural machine translation (NMT) model trained in table 5.6.

Epochs	Error%
5	0.2
6	5.7
7	4.7
8	5
9	5.7
10	4.7
11	4.9
12	5.2
13	5.9
14	4.9
15	5
16	4.6
17	5.3
18	4.9
19	4.9
20	4.9
21	4.9
22	5
23	5
24	4.5

Table 5.5: Error Rates of NMT models trained using the best trained OCR model best predictions for the characters in images of the whole training dataset. The best OCR model mentioned is the best among OCR models trained on the whole training dataset.

Dataset	Error%
Validation Dataset	4.5
Testing Dataset	4.707

Table 5.6: Error Rates of the best OCR model trained on the first half of the training data set on the validation testing data sets.

5.4 Half-based Pipeline

In this section I will show the error rates of the last OCR models trained on half of the training dataset with respect to the validation dataset 5.4. I have chosen the model trained for 62 epochs which obtained an error rate of **6.197%**. I also show the error rate of the chosen model on the training and testing datasets 5.7.

Lines	Epochs	Error%
75000	56	6.754
76000	57	6.609
77000	58	6.899
78000	58	6.855
79000	59	6.375
80000	60	6.609
81000	61	6.576
82000	61	6.375
83000	62	6.197
84000	63	6.721
85000	64	7.98
86000	64	7.3
87000	65	6.353
88000	66	7.323
89000	67	6.832
90000	67	6.788
91000	68	6.587

Table 5.7: Error Rates of the last OCR Models trained on the first half of the training data set on the validation data set.

Dataset	Error%
Training Dataset	4.988
Validation Dataset	6.197
Testing Dataset	6.744

Table 5.8: Error Rates of the best OCR model trained on the first half of the training data set on the validation testing data sets.

Next I show the error rate of the NMT Models trained using the chosen OCR model prediction on the **whole** training dataset with respect to the validation dataset 5.9. Here I have chosen trained for **55 epochs** for obtaining an error rate of **9.17%**. I show this models error rate on the testing dataset in table 5.10.

Epochs	Error%
51	9.28
52	9.34
53	9.42
54	9.34
55	9.17
56	9.5
57	9.2
58	9.18
59	9.2
60	9.36

Table 5.9: Error Rates of NMT models trained using the best trained OCR model prediction on the whole training dataset. The best OCR model mentioned is the best among OCR models trained on the first half of the training dataset.

Dataset	Error%
Validation Dataset	9.17
Testing Dataset	5.118

Table 5.10: Error Rates of the best NMT model trained on the whole training datset on the validation testing datasets.

I also show the error rate of the NMT models trained on just he **second half** of the training dataset with respect to the validation dataset in table 5.11. I also show the testing error rate of the model trained for **59 epochs** as it got the least error rate on the validation dataset 5.12.

Epochs	Error%
55	6.61
56	6.3
57	6.03
58	6.1
59	5.97
60	6.42
61	6.18
62	5.99
63	6.01
64	6.28
65	6.17
66	6.31
67	6.23
68	6.28
69	6.13
70	6.08

Table 5.11: Error Rates of NMT models trained using the best trained OCR model prediction on the second half of the training dataset. The best OCR model mentioned is the best among OCR models trained on the first half of the training dataset.

Dataset	Error%
Validation Dataset	5.97
Testing Dataset	7.075

Table 5.12: Error Rates of the best NMT model trained on the second half of the training datset on the validation testing datasets.

5.5 One-third-based Pipeline

In this section I will show the results regarding the OCR trained on **1/3** of the training dataset and the NMT trained after the best chosen OCR model on the other **2/3** of the dataset and on the whole training dataset. The validation error rates of the last OCR models trained are shown in table 5.13.

Lines	Epochs	Error%
75000	84	7.434
76000	85	7.144
77000	86	7.78
78000	88	8.627
79000	89	7.39
80000	90	7.724
81000	91	7.624
82000	92	7.724
83000	93	7.624
84000	94	7.847
85000	95	7.1
86000	97	6.765
87000	98	7.434
88000	99	8.214
89000	100	7.501
90000	101	7.78
91000	102	7.245

Table 5.13: Error Rates of the last OCR Models trained on the first one third of the training data set on the validation data set.

Here I have chosen the model trained on 97 epochs, as it showed the best error rate on the validation dataset **6.765%**. I also show the error rate of this model on the training and testing datasets in table 5.14.

Dataset	Error%
Training Dataset	4.632
Validation Dataset	6.765
Testing Dataset	7.437

Table 5.14: Error Rates of the best OCR model trained on the first one third of the training data set on the validation testing data sets.

Also, following is the validation error rates of the last NMT models trained using the chosen model's prediction on the whole training dataset 5.15. Also I show the result of the best NMT model when used to predict the output of the testing dataset in 5.16.

Epochs	Error%
54	6.2
55	6.3
56	6.3
57	6.2
58	6.5
59	6.7
60	10.7
61	6.4
62	6.6
63	6.5
64	6.3
65	6.4
66	6.3
67	6.3
68	6.3
69	6.5
70	6.3

Table 5.15: Error Rates of NMT models trained using the best trained OCR model prediction on the whole training dataset. The best OCR model mentioned is the best among OCR models trained on the first one third of the training dataset.

Dataset	Error%
Validation Dataset	5.4
Testing Dataset	5.339

Table 5.16: Error Rates of the best NMT model trained on the whole training datset on the validation testing datasets.

The last thing I show in this section is the validation error rates of the NMT models trained on just the second two-thirds of the training dataset in 5.17 and I show the error rate of the best NMT model on the testing dataset in 5.18.

Epochs	Error%
51	6.4
52	6.3
53	6.5
54	6.2
55	6.3
56	6.3
57	6.2
58	6.5
59	6.7

Table 5.17: Error Rates of NMT models trained using the best trained OCR model prediction on the second two thirds of the training dataset. The best OCR model mentioned is the best among OCR models trained on the first one third of the training dataset.

Dataset	Error%
Validation Dataset	6.2
Testing Dataset	6.294

Table 5.18: Error Rates of the best NMT model trained on the second two thirds of the training datset on the validation testing datasets.

5.6 Two-Thirds-based Pipeline

In this section I show the results of the OCR NMT models produced when the OCR model had **2 / 3** of the training dataset to be used for training. I first show the validation error rates of the last OCR models trained along with the training testing error rates of the best chosen model in 5.19 and 5.23.

Lines	Epochs	Error%
75000	45	7.234
76000	45	7.289
77000	46	6.71
78000	47	7.055
79000	47	6.487
80000	48	6.643
81000	48	6.788
82000	49	6.844
83000	49	6.23
84000	50	6.721
85000	51	9.875
86000	51	6.442

Table 5.19: Error Rates of the last OCR Models trained on the first two thirds of the training data set on the validation data set.

Dataset	Error%
Training Dataset	4.967
Validation Dataset	6.23
Testing Dataset	6.574

Table 5.20: Error Rates of the best OCR model trained on the first two thirds of the training data set on the validation testing data sets.

I also show the results for the NMT models trained using the chosen OCR model's prediction on the whole training dataset in 5.21.

Epochs	Error%
46	5.32
47	5.97
48	6.1
49	5.79
50	4.92
51	5.46
52	5.6
53	5.96
54	5.32
55	6.06

Table 5.21: Error Rates of NMT models trained using the best trained OCR model prediction on the whole training dataset. The best OCR model mentioned is the best among OCR models trained on the first two thirds of the training dataset.

Here I have chosen the model trained on **50** epochs which had an error rate of **4.92%** on the validation dataset. This model produced an error rate of **5.077%** on the testing dataset.

The last result I show in the section is the result of the NMT models trained using the prediction of the OCR model on the second **1/3** portion of the training dataset on the validation dataset in 5.22.

Epochs	Error%
60	5.49
61	6.19
62	5.92
63	6.1
64	4.74
65	5.31
66	5
67	4.79
68	5.37
69	6.01
70	5.79

Table 5.22: Error Rates of NMT models trained using the best trained OCR model prediction on the second one third part of the training dataset. The best OCR model mentioned is the best among OCR models trained on the first two thirds of the training dataset.

Dataset	Error%
Validation Dataset	4.74
Testing Dataset	6.942

Table 5.23: Error Rates of the best NMT model trained on the second one third of the training data set prediction by the best OCR model trained on the first two thirds of the training data set on the validation & testing data sets.

5.7 Significance Test

In this section, I will show how did I measure the significance of the best performing networks which used Neural Machine Translation (NMT) for error correction (One-based) and I will show whether the results are significantly different or not.

To measure the significance of the best performing networks I used the **Paired Samples T Test**.

Our Null Hypothesis is that the means of the error percentage before and after error correction is the same.

To calculate the t-score I used the following formula:

$$\frac{\frac{\sum D}{N}}{\sqrt{\frac{\sum D^2 - (\frac{(\sum D)^2}{N})}{(N-1)N}}}$$

- N is the number of items in the sample which is the number of images used for testing in our case.
- $\sum D$ is the sum of the differences of scores of the two methods -which are OCR without error correction and OCR with error correction- on the sample.
- $\sum D^2$ is the sum of the square of differences of scores of the two methods -which are OCR without error correction and OCR with error correction- on the sample.

T score in our case is **14.207**. I have chosen an alpha level of **0.05**, which gets a p-value of **1.658** which is less than our T score. So in this case I can reject the null hypothesis.

Chapter 6

Conclusion

6.1 Results Summary

In this section I show a summary of the results in figure 6.1. For the one-third based partitioning I are showing a result of **5.339%** which is the error rate when the NMT model is trained on the whole training dataset, which is better than **6.294%** when the OCR is also trained on one third of the training dataset but the NMT is trained on just two thirds of the trianing dataset. For the half-based partioning I are showing the error rate of **5.118%** which is achieved when the NMT model is trained on the whole training dataset. Also, for the two-thirds based partitioning I are showing the result of **5.077%** which is much better than **6.942%** obtained when the NMT model is trained on the other third portion of the training dataset.

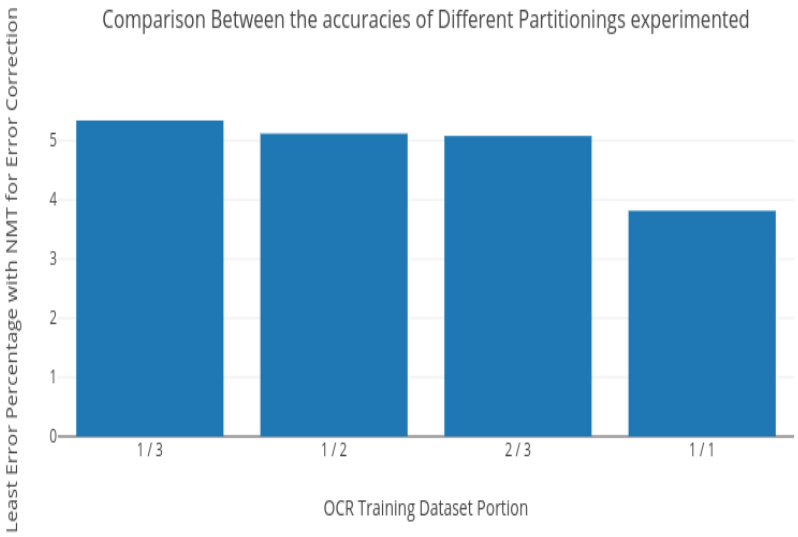

Figure 6.1: Comparison Between the Error Rates of the Different Dataset Partitionings.

6.2 Interpretation

As I showed in the results, optical character recognition (OCR) combined with neural machine translation (NMT) for error correction always achieves better accuracy than optical character recognition (OCR) alone in all of the different partitionings.

I also showed that it is better to train the error correction models on the dataset on which the optical character recognition (OCR) models are trained. The best accuracy for the pipeline is obtained when the optical character recognition (OCR) models and the neural machine translation (NMT) models are trained on the whole training dataset which is **3.814%**.

Chapter 7

Future Work

As I showed that the results of the Optical character recognition (OCR) with error correction using neural machine translation (NMT) are always better than the results of the OCR when trained and used without post error correction. This is because of what the NMT model learns about the behavior and the mistakes of the OCR model in some sequences and patterns. Until this point, the NMT models are trained using the OCR output on some images along with the ground truth of these images. Still, the neural machine translation (NMT) models trained doesn't know much about the source images themselves. I suggest going through some approach similar to the neural machine translation (NMT) approach for error correction that includes the images in the input for training the error correction neural network. I show our insight about the difference in the second stage of the pipeline between the approach I experimented and the approach I are suggesting in 7.1. I see that connecting information of the image with the OCR prediction on this image along with the ground truth can show much better results than learning information about the OCR output and the ground truth alone.

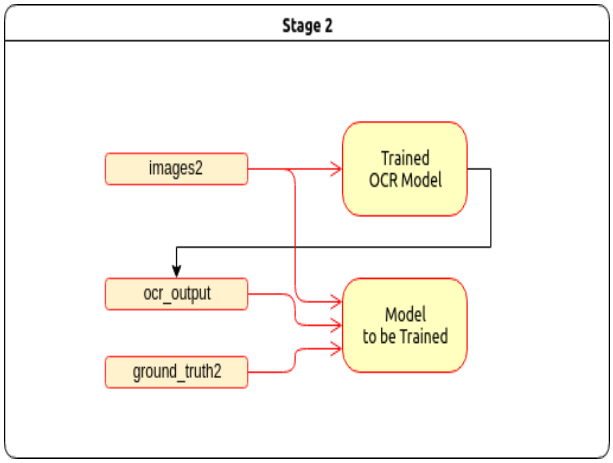

Figure 7.1: Showing our insight for how can future work proceed after adding the source images to the arguments of the neural machine translation (NMT) input for training.

Appendix

Appendix A

Lists

List of Figures

2.1 Ocropus Workflow of the separate command line tools from OCRopus. ocropus-econf, ocropus-rtrain ocropus-rpred are the 3 commands used through our experiments. 6

2.2 Rule Based Machine Translation Methods. 7

2.3 Encoder-Decoder Neural Network Architecture. 8

3.1 Word Based Sequence to Sequence model examples illustrating how are the words should be separated from each other as well as from other punctuation characters. 13

3.2 Stage one of OCR Error Correction showing how does the OCR models use the training dataset specified for OCR to be trained on 14

3.3 Second stage of OCR Error Correction, showing how does the neural machine translation (NMT) models train using the best trained optical character recognition (OCR) model's prediction on the training dataset specified for the neural machine translation (NMT) network. 15

3.4 OCR Error Correction Predicton Steps showing how does the best trained optical character recognition (OCR) model is used with the best trained neural machine translation (NMT) model for predicting input text. . . . 15

4.1 Latin Book Page. 17

4.2 Latin Book Page. 18

4.3 A sample of the dataset I are working on. Showing two Cropped and Binarized Parts from the Latin Book along with their corresponding digitized ground truth. 18

4.4 Sample From the Latin Dataset images. 19

4.5 Problem with the latin dataset was having tabs before the beginning of the sentences in the ground truth files. 19

4.6 Old Set of Characters which is the OCRopus default character set. 21

4.7 New Set of Characters obtained by specifying to OCRopus to construct a codec from the training ground truth files. 21

4.8 Showing the two stages needed for training optical character recognition (OCR) models and neural machine translation (NMT) models based on the one-based partitioning. The optical character recognition (OCR) and the neural machine translation (NMT) models are both trained on the whole training dataset. 24

4.9 Half-based partitioning Diagram showing stages ones and two when the neural machine translation (NMT) models are trained using the prediction of the optical character recognition (OCR) model on the whole training dataset. 25

4.10 Half-based partitioning Diagram showing stages ones and two when the neural machine translation (NMT) models are trained using the prediction of the optical character recognition (OCR) model on the second half of the training dataset. 26

4.11 One-third-based partitioning Diagram showing stages ones and two when the neural machine translation (NMT) models are trained using the prediction of the optical character recognition (OCR) model on the whole training dataset. 27

4.12 One-third-based partitioning Diagram showing stages ones and two when the neural machine translation (NMT) models are trained using the prediction of the optical character recognition (OCR) model on the second two thirds of the training dataset. 27

4.13 Two-thirds-based partitioning Diagram showing stages ones and two when the neural machine translation (NMT) models are trained using the prediction of the optical character recognition (OCR) model on the whole training dataset. 28

4.14 Two-thirds-based partitioning Diagram showing stages ones and two when the neural machine translation (NMT) models are trained using the prediction of the optical character recognition (OCR) model on the second one third of the training dataset. 29

6.1 Comparison Between the Error Rates of the Different Dataset Partitionings. 45

7.1 Showing our insight for how can future work proceed after adding the source images to the arguments of the neural machine translation (NMT) input for training. 47

Bibliography

[1] Preeti Verma Abhishek Verma, Suket Arora. Ocr-optical character recognition. *Advance Research in Science and Engineering*, 2(5):181–182, 2016.

[2] Thomas Breuel. Announcing the ocropus open source ocr system. *Google Developers Blog*, page 1.

[3] Dharmendra Patel Chirag Patel, Atul Patel. Optical character recognition by open source ocr tool tesseract: A case study. *Computer Applications*, 55(10):50–51, 2012.

[4] Yaser AlOnaizan Jan Curin Michael Jahr Kevin Knight John Laerty Dan Melamed FranzJosef Och David Purdy Noah A Smith Davi. Statistical machine translation. *Final Report JHU Worksho*, page 2.

[5] Walter S. RosenbaumJohn J. Hilliard. System and method for deferred processing of ocr scanned mail. *IEEE Transactions on Intelligent Transportation Systems*, 7:1 – 2, 1991.

[6] Andreas Dengel Kareem Mokhtar, Syed Saqib Bukhari. Ocr error correction:state-of-the-art vs an nmt based approach. 2017.

[7] C.N.E. Anagnostopoulos ; I.E. Anagnostopoulos ; V. Loumos ; E. Kayafas. A license plate recognition algorithm for intelligent transportation system applications. *IEEE Transactions on Intelligent Transportation Systems*, 7:377 – 378, 2006.

[8] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. Opennmt: Open-source toolkit for neural machine translation. In *Proc. ACL*, 2017.

[9] Dzmitry Bahdanau Yoshua Bengio Kyunghyun Cho, Bart van Merrienboer. On the properties of neural machine translation: Encoderdecoderapproaches. *arXiv:1409.1259v2*, pages 1–2.

[10] Andreas Dengel Martin Jenckel, Syed Saqib Bukhari. anyocr: A sequence learning based ocr system for unlabeled historical documents. pages 1, 6, 2016.

[11] Alexandra Birch Chris Callison-Burch Marcello Federico Brooke Cowan Wade Shen Christine Moran Nicola Bertoldi Richard Zens Chris Dyer Ondej Bojar Alexandra Constantin Evan Herbst Philipp Koehn, Hieu Hoang. Moses: open source toolkit

for statistical machine translation. pages 177–180. 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions, 2007.

[12] Urmila Shrawankar Tejaswinee Wakde. A multilingual currency interpreter. *International Conference on Convergence of Technology - 2014*, page 4.