Generic Text Recognition using Long Short-Term Memory Networks

by

Adnan Ul-Hasan

Thesis approved by the

Department of Computer Science
University of Kaiserslautern
for the award of Doctoral Degree

Doctor of Engineering (Dr. -Ing)

Date of PhD Defense: 11.01.2016

Dean of the Department:
Prof. Dr. Klaus Schneider

Chairperson of the PhD Committee:
Prof. Dr. Paul Lukowicz

Thesis Reviewers:
Prof. Dr. Andreas Dengel, DFKI Kaiserslautern
Associate Prof. Dr. Faisal Shafait, SEECS, NUST Pakistan
apl. Prof. Dr. Marcus Liwicki, University of Kaiserslautern

D 386

It always seems impossible until it is done.

Nelson Mandela

Abstract

The task of printed Optical Character Recognition (OCR) is considered a “solved” issue by many Pattern Recognition (PR) researchers. The notion, however, partially true, does not represent the whole picture. Although, it is true that state-of-the-art OCR systems for many scripts exist, for example, for Latin, Greek, Han (Chinese), and Kana (Japanese), there is still a need for exhaustive research for many other challenging modern scripts. Example of such scripts are: cursive Nabataean, which include Arabic, Persian, and Urdu; and the Brahamic family of scripts, which contain Devanagari, Sanskrit, and its derivatives. These scripts present many challenging issues for OCR, for example, change in shape of character within a word depending upon its location, kerning, and a huge number of ligatures. Moreover, OCR research for historical documents still requires much probing; therefore, efforts are required to develop robust OCR systems to preserve the literary heritage.

Likewise, there is a need to address the issue of OCR of multilingual documents. Plenty of multilingual documents exist in the current time of globalization, which has increased the influence of different languages on each other. There is an increase in the usage of foreign words and phrases in articles, newspapers, and books, which are generating a large body of multilingual literature everyday. Another effect is seen in the products we use in our daily lives. From packaging of imported food items to sophisticated electronics, the demand of international customers to access information about these products in their native language is ever increasing. The use of multilingual operational manuals, books, and dictionaries motivates the need to have multilingual OCR systems for their digitization.

The aim of this thesis is to find the answers to some of these challenges using the contemporary machine learning methodologies, especially the Recurrent Neural Networks (RNN). Specifically, a recent architecture of these networks, referred to as Long Short-Term Memory (LSTM) networks, has been employed to OCR modern as well historical documents. The excellent OCR results obtained on these documents encourage us to extend their application to the field of multilingual OCR.

The LSTM networks are first evaluated on standard English datasets to benchmark their performance. They yield better recognition results than any other contemporary OCR techniques without using sophisticated features and language modeling. Therefore, their application is further extended to more complex scripts that include Urdu Nastaleeq and Devanagari. For Urdu Nastaleeq script, LSTM networks achieve the best reported OCR results (2.55% Character Error Rate (CER)) on a publicly available data set, while for Devanagari script, a new freely available database has been introduced on which CER of 9% is achieved.

The LSTM-based methodology is further extended to the OCR of historical documents. In this regard, this thesis focuses on Old German Fraktur script, medieval Latin script of the 15th century, and the Polytonic Greek script. LSTM-based systems outperform the contemporary OCR systems on all of these scripts. For old documents, it is usually very hard to prepare transcribed dataset for training a neural network in supervised learning paradigm. A novel methodology has been proposed by combining segmentation-based and segmentation-free approaches to OCR scripts for which no transcribed training data is available. For German Fraktur and Polytonic Greek scripts, artificially generated data from existing text corpora yield highly promising results (CER of <1% and 5% for Fraktur and Polytonic Greek scripts respectively).

Another major contribution of this thesis is an efficient Multilingual OCR (MOCR) system, which has been achieved in two ways. Firstly, a sequence-learning based script identification system has been developed that works at the text-line level; thereby, eliminating the need to segment individual characters or words prior to actual script identification. And secondly, a unified approach for dealing with different types of multilingual documents has been proposed. The core motivation behind this generalized framework is the reading ability of the human mind to process multilingual documents, where no script identification takes place. In this design, the LSTM networks recognize multiple scripts simultaneously without the need to identify different scripts. The first step in building this framework is the realization of a language independent OCR system that recognizes multilingual text in a single step, using a single Long Short-Term Memory (LSTM) model. The language independent approach is then extended to a script independent OCR framework that can recognize multiscript documents using a single OCR model. The proposed generalized approach yields low error rate (1.2%) on a database of English-Greek bilingual documents.

In summary, this thesis aims to extend the OCR research, from modern Latin scripts to old Latin, to Greek and to other “underprivileged” scripts such as Devanagari and Urdu Nastaleeq. It also provides a generalized OCR framework in dealing with multilingual documents.

Contents

Abstract v

List of Figures xi

List of Tables xv

Abbreviations xvii

1 Introduction 1
- 1.1 Motivation . . . 4
- 1.2 Research Hypothesis . . . 6
- 1.3 Contributions of the Thesis . . . 6
- 1.4 Thesis Structure . . . 7

2 Research Methodologies for Printed OCR 11
- 2.1 Segmentation-Based OCR . . . 12
  - 2.1.1 Template Matching for OCR . . . 12
  - 2.1.2 Over Segmentation Methods . . . 14
- 2.2 Segmentation-Free OCR . . . 14
  - 2.2.1 HMM-Based OCR Approach . . . 16
  - 2.2.2 Sequence Learning Approach . . . 16

Part – I : Challenges for Printed OCR

3 Challenges for Robust OCR 19
- 3.1 Modern English . . . 20
  - 3.1.1 Challenges for OCR in Printed English . . . 20
  - 3.1.2 OCR Databases for modern English . . . 21
- 3.2 Latin Script of Late Medieval Ages . . . 22
  - 3.2.1 Challenges for Old Latin OCR . . . 22
  - 3.2.2 OCR Databases for Old Latin . . . 23
- 3.3 German Fraktur Script . . . 23
  - 3.3.1 Challenges for Fraktur OCR . . . 24
  - 3.3.2 OCR Databases for Fraktur . . . 24
- 3.4 Polytonic Greek Script . . . 25
  - 3.4.1 OCR Challenges for Polytonic Greek . . . 25
  - 3.4.2 OCR Databases for Polytonic Greek . . . 25

3.5 Devanagari Script 26
  3.5.1 Challenges for Devanagari OCR 27
  3.5.2 OCR Databases for Devanagari 27
3.6 Urdu Nastaleeq Script 27
  3.6.1 Challenges for Nastaleeq OCR 28
  3.6.2 Databases for Urdu Nastaleeq OCR 33
3.7 Printed Documents with Multiple Scripts and Languages 35
  3.7.1 Categorization of Multilingual Documents 36
  3.7.2 OCR Datasets for Multilingual Documents 36
3.8 Chapter Summary 38

4 Benchmark Datasets for OCR 39
4.1 Related Work 40
4.2 Semi-automated Database Generation 41
  4.2.1 Preprocessing 42
  4.2.2 Ligature Extraction 42
  4.2.3 Clustering 43
  4.2.4 Ligature Labeling 44
  4.2.5 Experiments and Results 45
  4.2.6 Conclusion 46
4.3 Synthetic Text-Line Generation 46
4.4 OCR Databases 47
  4.4.1 Deva-DB – OCR Database for Devanagari Script 47
  4.4.2 Polyton-DB – Greek Polytonic Database 49
  4.4.3 Databases for Multilingual OCR (MOCR) 49
4.5 Chapter Summary 52

Part – II : OCR for Monolingual Documents

5 OCR for Printed Modern Scripts 53
5.1 Design of LSTM-Based OCR System 53
  5.1.1 Network Parameters Selection 55
  5.1.2 Features 57
  5.1.3 Performance Metric 57
5.2 Printed English OCR 57
  5.2.1 Related Work 58
  5.2.2 Database 59
  5.2.3 Experimental Evaluation and Results 59
  5.2.4 Error Analysis 62
  5.2.5 Conclusions 62
5.3 Printed Devanagari OCR 63
  5.3.1 Related Work 63
  5.3.2 Database 64
  5.3.3 Experimental Evaluation and Results 65
  5.3.4 Error Analysis 66
  5.3.5 Conclusions 68
5.4 Printed Urdu Nastaleeq OCR 68

- 5.4.1 Related Work . . . 70
- 5.4.2 Database . . . 70
- 5.4.3 1D-LSTM Networks for Urdu Nastaleeq Script . . . 71
- 5.4.4 Error Analysis . . . 74
- 5.4.5 HSLSTM Networks for Urdu Nastaleeq OCR . . . 75
- 5.4.6 Comparative Analysis . . . 76
- 5.4.7 Conclusion . . . 77
- 5.5 Chapter Summary . . . 77

6 OCR for Historical Documents 79
- 6.1 Fraktur and Polytonic Greek Scripts . . . 80
  - 6.1.1 Related Work . . . 80
  - 6.1.2 Database . . . 81
  - 6.1.3 Experimental Evaluation for Fraktur Script . . . 82
  - 6.1.4 Experimental Evaluation for Polytonic Greek Script . . . 82
  - 6.1.5 Conclusion . . . 85
- 6.2 OCRoRACT: A Sequence Learning OCR System Trained on Isolated Characters . . . 86
  - 6.2.1 Introduction . . . 86
  - 6.2.2 Methodology . . . 87
  - 6.2.3 The Database . . . 89
  - 6.2.4 The System Parameters . . . 89
  - 6.2.5 Results . . . 90
  - 6.2.6 Error Analysis and Discussions . . . 91
  - 6.2.7 Conclusion . . . 93
- 6.3 Chapter Summary . . . 94

$Part - III : OCR\, for\, Multilingual Documents$

7 Sequence Learning for Multiple Script Identification 95
- 7.1 Scope . . . 96
- 7.2 Heuristic-Based Approaches for Script Identification . . . 96
- 7.3 Machine Learning Based Approaches . . . 98
- 7.4 LSTM Networks for Script Identification . . . 99
- 7.5 Experiments . . . 100
  - 7.5.1 LSTM Networks' Architecture . . . 100
  - 7.5.2 Database . . . 101
  - 7.5.3 Performance Metric . . . 101
- 7.6 Results and Discussions . . . 101
- 7.7 Chapter Summary . . . 103

8 Generalized OCR Framework for Multilingual Documents 105
- 8.1 Traditional Approaches for MOCR . . . 108
- 8.2 Language Independent OCR with LSTM Networks . . . 108
  - 8.2.1 Experiment Setup . . . 109
  - 8.2.2 Preprocessing . . . 110
  - 8.2.3 Database . . . 110

- 8.2.4 LSTM Architecture and Parameters . . . 110
- 8.2.5 Results . . . 111
- 8.2.6 Error Analysis . . . 113
- 8.2.7 Conclusion . . . 115
- 8.3 Generalized OCR with LSTM Networks . . . 115
  - 8.3.1 Preprocessing . . . 117
  - 8.3.2 Database . . . 117
  - 8.3.3 LSTM Architecture and Parameters . . . 117
  - 8.3.4 Results . . . 119
  - 8.3.5 Error Analysis . . . 119
- 8.4 Chapter Summary . . . 121

9 Conclusions and Future Work 125

A Long Short-Term Memory Networks 129
- A.1 LSTM Networks . . . 129
  - A.1.1 Connectionist Temporal Classification (CTC) . . . 132
- A.2 LSTM Architectures . . . 134
  - A.2.1 Single Layer LSTM Network . . . 134
  - A.2.2 Multilayer LSTM Network . . . 134
  - A.2.3 Hierarchical SubSampling LSTM (HSLSTM) Networks . . . 135
  - A.2.4 Bidirectional LSTM Networks (BLSTM) . . . 136
  - A.2.5 One Dimensional LSTM (1D-LSTM) Networks . . . 136
  - A.2.6 Multidimensional LSTM (MDLSTM) Networks . . . 137
- A.3 Tunable Parameters . . . 137

B Text-Line Normalization 141
- B.1 Image Rescaling . . . 141
- B.2 Zone-Based Normalization . . . 142
- B.3 Token-Dictionary based Normalization . . . 142
- B.4 Filter-based Normalization . . . 144

Bibliography 147

List of Figures

1.1 The Document Image Analysis (DIA) pipeline . . . 3
1.2 Three categories of documents that are hindering the implementation of a robust OCR technology . . . 5
1.3 Thesis structure and recommended flow of reading . . . 8

2.1 Illustration of basic template matching . . . 13
2.2 Illustration of segmentation process using over-segmentation method . 15
2.3 Figure showing the basic unit in HMM-based OCR . . . 16
2.4 Illustration of how RNNs are used for the OCR task . . . 18

3.1 Sample text-line images from UW-III database . . . 21
3.2 Character set used in the old Latin documents . . . 22
3.3 Example of Fraktur script . . . 23
3.4 Shape confusion in Fraktur script . . . 24
3.5 Document quality degradation caused during preprocessing . . . 24
3.6 Word formation in Devanagari script . . . 26
3.7 Reading direction in Nastaleeq script . . . 28
3.8 Diagonal writing flow in Nastaleeq . . . 29
3.9 Comparison between Nastaleeq and Naskh scripts . . . 29
3.10 The issue of characters overlapping within a ligature or between two adjacent ligatures. . . . 29
3.11 A word in Urdu with its constituent characters. . . . 30
3.12 The shape of a character at different positions in a word . . . 30
3.13 The issue of contextual shape change for a character . . . 31
3.14 Issue of random baseline in Nastaleeq . . . 31
3.15 Some examples of Urdu ligatures . . . 32
3.16 Importance of dots and diacritics . . . 33
3.17 Effect of close proximity on the location of dots . . . 33
3.18 Urdu document scribed by a *Katib* . . . 34
3.19 Three main categories of multilingual documents . . . 37

4.1 Examples of ligature-based Urdu Nastaleeq script . . . 42
4.2 The graphical interface to examine the clusters and to assign the unique identification . . . 44
4.3 The synthetic text-line image generation process using various degradation parameters . . . 47
4.4 Four sample images taken from the training set . . . 48
4.5 Sample documents from three main sources in Polyton-DB . . . 50
4.6 Some samples from synthetically generated images . . . 51

5.1 The complete pipeline for an LSTM-based OCR system 54
5.2 Recognition error versus hidden layer sizes 55
5.3 Training time as a function of hidden layer size 56
5.4 Recognition error versus various learning rates 56
5.5 Configuration of trained LSTM network 60
5.6 An illustration of the training steps of the LSTM line recognizer 61
5.7 Comparison of error rates of the LSTM recognizer with Tesseract, ABBYY, and the older version of OCRopus 61
5.8 Input-output pairs from the LSTM line recognizer from modern English . 62
5.9 Samples taken from the real scanned set of line images where LSTM-based line recognizer fails 65
5.10 Categorization of Urdu characters 69
5.11 Sample text-line images from UPTI database 71
5.12 Training pipeline for Urdu Nastaleeq OCR 72
5.13 CER during the training phase 73
5.14 Input/output from the LSTM-base OCR illustrating capabilities and errors 74
5.15 Comparison of CER for various values of normalized height 76

6.1 Comparison of error rates of the LSTM recognizer with Tesseract, and ABBYY 82
6.2 CER of LSTM in the first experiment for various training iterations 83
6.3 CER of LSTM in the second experiment for various iterations 84
6.4 Comparison of three OCR systems for Polytonic Greek 84
6.5 The process diagram of OCRoRACT system 88

7.1 Complete pipeline to use LSTM networks for multiple script identification 99
7.2 The location information predicted by the trained LSTM model on a test text-line image 100
7.3 Samples of some text-lines separated by LSTM-based script identification method 103

8.1 The traditional approach used to OCR multilingual documents 106
8.2 The two different approaches to OCR multilingual documents 115
8.3 Some examples from the training data used for generic OCR 118
8.4 The network architecture used for the Generalized OCR methodology . 118
8.5 Training errors for LSTM network 119
8.6 An example of correctly recognized text-line image 120
8.7 A sample text-line image with incorrectly predicted labels 121

A.1 Illustration of a basic LSTM memory cell 130
A.2 RNN where neurons at the hidden layer form self-loops for feedback . . 134
A.3 Multilayer LSTM network with ‘n’ hidden layers 135
A.4 Schematic diagram of a subsampling hierarchical network 136
A.5 The information flow in a BLSTM network 137
A.6 The process of converting a text-line image into a 1D sequence 138

B.1 Zone-based normalization for Devanagari text-line image 142
B.2 Extraction of x-height and baseline of a text-line in The process of converting a text-line image into a 1D sequence 143

B.3 Filter-based method for script-independent text-line normalization . . . 145
B.4 Samples text-line images normalized using filter-based method of text-line normalization . . . 145

List of Tables

4.1 Comparison of the character frequency of proposed training database with already published statistical results. 48
4.2 Number of text-line images in each of English, French, German and mix-script datasets. 51

5.1 Results of the experiment 66
5.2 Confusion matrix for Tesseract 67
5.3 Comparison of various HSLSTM architectures 76
5.4 Comparison of various LSTM-based OCR systems on UPTI database . . . 77

6.1 Most frequent errors of the LSTM-recognizer 85
6.2 Quantitative comparison between Tesseract, OCRopus and OCRoRACT . 91
6.3 Qualitative comparison between Tesseract, OCRopus and OCRoRACT . 91
6.4 Comparison of top confusions of three systems for ‘T1’ dataset. 92
6.5 Comparison of top confusions of three systems for ‘T2’ dataset. 93

7.1 Top 15 confusions for script identification task. 102

8.1 Experimental results of applying LSTM networks for MOCR 111
8.2 Sample outputs from four LSTM networks trained for English, German, French and Mixed-Data 112
8.3 Top confusions for applying LSTM models for various tasks 114
8.4 Confusion matrix of the Generalized OCR approach 122
8.5 Top confusions from generalized OCR methodology considering the neighboring context. 122

Abbreviations

RNN Recurrent Neural Networks

LSTM Long Short-Term Memory

CNN Convolutional Neural Networks

MOCR Multilingual OCR

OCR Optical Character Recognition

DIA Document Image Analysis

PR Pattern Recognition

ML Machine Learning

MDLSTM Multidimensional LSTM

HSLSTM Hierarchical Subsampling LSTM

BLSTM Bidirectional LSTM

1D-LSTM One Dimensional LSTM

RTL Right-to-Left

ANN Artificial Neural Networks

CTC Connectionist Temporal Classification

HMM Hidden Markov Models

APTI Arabic Printed Text Images

UPTI Urdu Printed Text Images

IfN	Institute for Communications Technology
ENIT	Ecole Nationale d'Ingénieur de Tunis
MLP	Multilayer Perceptron
SVM	Support Vector Machines
PCA	Principal Component Analysis
CER	Character Error Rate
UW3	University of Washington-III
tanh	Tangent Hyperbolic
GT	Ground-Truth
DBN	Deep Belief Networks
CV	Computer Vision

To my parents
To my wife

Chapter 1

Introduction

Research in Optical Character Recognition (OCR) is an old field in Pattern Recognition (PR). The process of human reading is the inspiration behind developing a machine that could read text with the same proficiency as humans do. In fact, the very early systems reported for OCR were intended to help the blind to read. Decades of research in this field has resulted in many practical systems, from sorting posts in post offices [RH89] to automatically recognizing bank checks [GAA+99], from handheld scanners to sophisticated systems reading the text in natural scenes [Mat15].

The process of converting paper documents into their digital counterparts is commonly knows as **Document Image Analysis (DIA)**. The pipeline of a typical DIA system is shown in Figure 1.1. The OCR module in a DIA recognizes text; however, it cannot readily do so because of the presence of graphics, images, page orientation and skew issues. It is therefore necessary to separate text from the non-text regions in a given document image, and bring it to a form where the OCR module can be applied.

The first step in many DIA systems is to binarize the input image. This basically segments the document image into foreground and background pixels [Gat14]. This not only lowers the storage requirements and simplifies further processing but also cleans the document from unwanted details, for instance, noise, bleed-, or shine-through. The task of layout analysis in a DIA system is to extract text-lines, words, or characters from any given scanned page. It is very important to do the layout analysis step with high accuracy; otherwise, the efficiency of the recognition step would be compromised.

The core of many DIA systems is the OCR module. This module converts the document from image form to the corresponding textual form, which is suitable for information extraction, retrieval, and indexing. Over decades of research in OCR, various methods have been proposed to carry out the text recognition task. As technology becomes efficient and economical, OCR algorithms have begun to be used for more and more complex languages and scripts. These new scripts bring specific challenges to the table, thus resulting in new and improved methodologies.

Over the past decade, Hidden Markov Models (HMM) have received a lot of attention, mainly in the field of speech and handwriting recognition. They have been used for OCR of various printed scripts as well (see [MSLB98, EM08, HHK08]). Though HMMs yield good results, they suffer from a known property of generative models, that is, the assumption about the probability of each individual state is independent of its previous states. Moreover, there are many important heuristics that one has to take care of, like *HMM structures*, *input features*, and *character segmentation algorithms*.

In order to overcome these shortcomings of pure HMM-based recognition methods, researchers have proposed HMM/ANN hybrid approaches [Ras14]. In these approaches, Artificial Neural Networks (ANN) are used as discriminative feature extractors, while the HMM is used as a text recognition engine. Although, these hybrid approaches perform better than solely HMM-based ones; they still suffer from the undesired properties of HMMs [Gra12].

Recurrent Neural Networks (RNN) are basically an extension of the feedforward neural networks. The difference lies in the way neurons at the hidden layer(s) are connected. At any given time, a neuron in a hidden layer receives input not only from the external input state, but also from the previous hidden state. RNNs are considered good at context-aware processing and at recognizing patterns occurring in time-series [Sen94]. However, traditional RNNs have not shown competitive performance in large scale tasks like OCR and speech recognition, perhaps due to the *vanishing gradient problem* [HBFS01, BSF94]. This problem is explained in detail in Section 2.2.2.

The Long Short-Term Memory (LSTM) [HS97] architecture was designed to overcome this issue. It is a highly non-linear recurrent neural network with multiplicative “gates” and additive feedback. Graves et al. [GLF$^+$08] introduced Bidirectional LSTM (BLSTM) architecture for accessing context in both forward and backward direction. Both layers are connected to a single output layer. To avoid the requirement of segmented training data, Graves et al. [GFGS06] used a forward-backward algorithm (referred to as the Connectionist Temporal Classification (CTC) in the literature) to align transcripts with the output of the neural network.

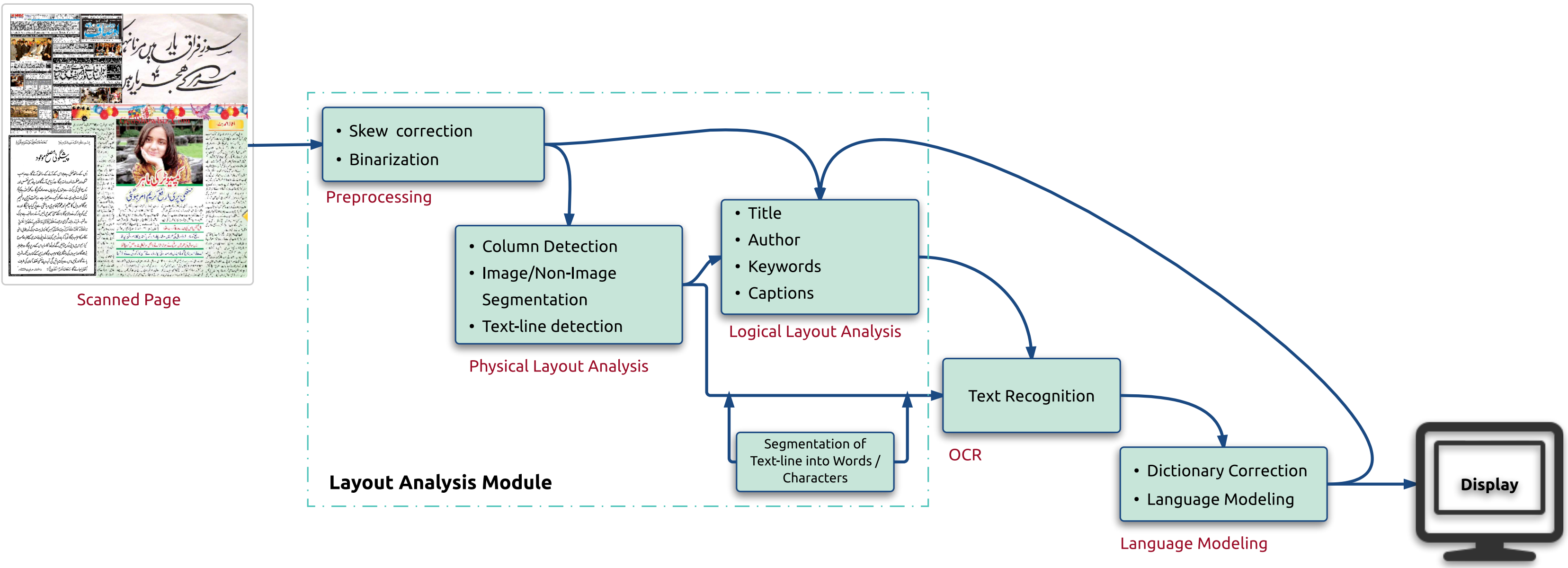

Scanned Page
Skew correction
Binarization
Preprocessing
Column Detection
Image/Non-Image Segmentation
Text-line detection
Physical Layout Analysis
Title
Author
Keywords
Captions
Logical Layout Analysis
Segmentation of Text-line into Words / Characters
Layout Analysis Module
Text Recognition
OCR
Dictionary Correction
Language Modeling
Language Modeling
Display

Figure 1.1: The Document Image Analysis (DIA) pipeline. A DIA consists of three main modules. The first major step is to segment the textual information from a given scanned page. The layout analysis module extracts text-lines, words or characters from a given page and the OCR module does the actual text recognition. Errors induced by these two modules are then corrected at a post-processing step using language models or dictionary correction.

An important feature of LSTM-based line recognizers is their ability to achieve very high recognition accuracy without using any sophisticated, handcrafted, or specialized features and without the use of any kind of post-processing steps, such as language modeling, dictionary correction, or any other adaptation. This makes the whole process very easy to use and applicable to a wide variety of scripts and languages.

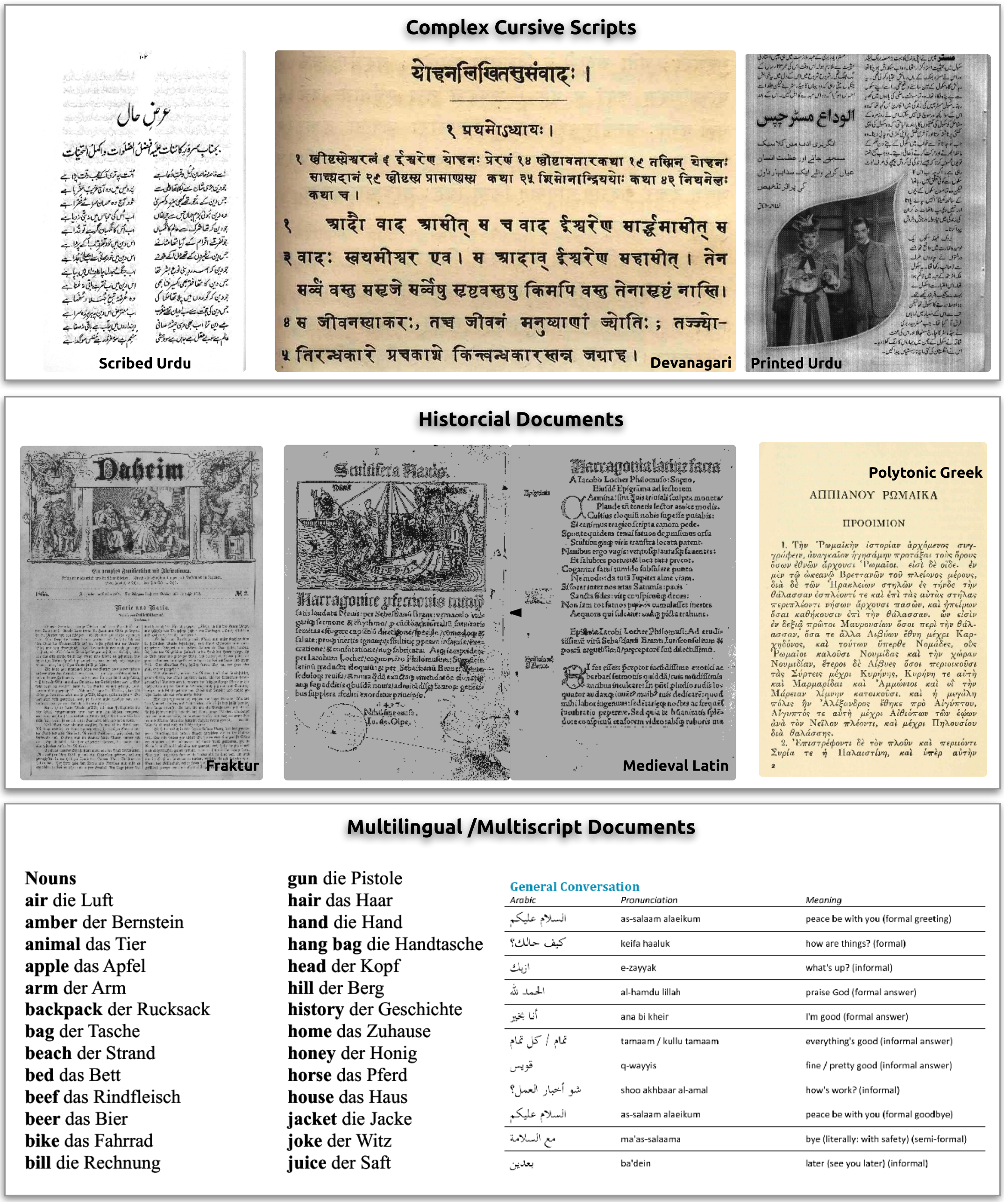
1.1 Motivation

Despite the technological advancements in the field of OCR systems, the practical use of such systems is often unsatisfactory. Variations in capturing techniques directly affect the final recognition of these automated systems. Scanning artifacts result in many degradations, such as connected characters, ink spread, bleed-, and shine-through. The case of digitizing historical documents gives rise to several other challenges, such as torn pages, archaic orthographies, and the use of ancient scripts. The complexity of scripts to be recognized plays a limiting factor, not only in historical scripts but also for many modern scripts where complex grapheme structure and difficult orthographic rules challenge the capabilities of modern recognition systems.

Another issue that has been reported more frequently in recent years is the case of multilingual documents, which have resulted from globalization. Operational manuals, multilingual books, and translations are some of the main sources of multilingual documents. Dealing with multiple grapheme structures of various scripts is a challenge for OCR systems, which have been designed for single-script documents. These multiple challenges are outlined in Figure 1.2, with representative images to clarify the issues.

OCR research for many of the above-mentioned challenges is still in its infancy. Many complex, modern scripts, such as Devanagari and Arabic-like scripts, lack a reliable OCR system. Same is the case with historical documents, where unavailability of Ground-Truth (GT) data is a major limitation in developing automatic recognition systems.

This thesis aims to solve some of these problems by applying state-of-the-art RNNs on one hand and by developing novel techniques on the other to address the issue of limited transcribed data. RNN-based methods have yielded excellent OCR results. The added advantage of using automatically-learned features and simple usage allow the same architecture to be used for a variety of scripts with outstanding results.


Complex Cursive Scripts
Scribed Urdu
Devanagari
Printed Urdu
Historcial Documents
Fraktur
Medieval Latin
Polytonic Greek
Multilingual /Multiscript Documents

عرض حال

الوداع مسٹر چپس

योहनलिखितसुसंवादः ।

१ प्रथमोऽध्यायः ।

१ ख्रीष्टस्येश्वरत्वं ६ ईश्वरेण योहनः प्रेरणं १४ ख्रीष्टावतारकथा १९ तस्मिन् योहनः साक्ष्यदानं २९ ख्रीष्टस्य प्रामाण्यस्य कथा ३५ शिमोनान्द्रिययोः कथा ४३ नियनेलः कथा च ।

१ आदौ वाद आसीत् स च वाद ईश्वरेण सार्द्धमासीत् स
२ वादः स्वयमीश्वर एव। स आदाव् ईश्वरेण सहासीत्। तेन
सर्व्वं वस्तु ससृजे सर्व्वेषु सृष्टवस्तुषु किमपि वस्तु तेनासृष्टं नास्ति।
४ स जीवनस्याकरः, तच्च जीवनं मनुष्याणां ज्योतिः; तज्ज्यो-
५ तिरन्धकारे प्रचकाशे किन्त्वन्धकारस्तन्न जग्राह।

Daheim

ΑΠΠΙΑΝΟΥ ΡΩΜΑΙΚΑ

ΠΡΟΟΙΜΙΟΝ

1. Τὴν Ῥωμαϊκὴν ἱστορίαν ἀρχόμενος συγγράφειν, ἀναγκαῖον ἡγησάμην προτάξαι τοὺς ὅρους ὅσων ἐθνῶν ἄρχουσι Ῥωμαῖοι. εἰσὶ δὲ οἵδε· ἐν μὲν τῷ ὠκεανῷ Βρεττανῶν τοῦ πλείονος μέρους, διὰ δὲ τῶν Ἡρακλείων στηλῶν ἐς τήνδε τὴν θάλασσαν ἐσπλέοντί τε καὶ ἐπὶ τὰς αὐτὰς στήλας περιπλέοντι νήσων ἄρχουσι πασῶν, καὶ ἠπείρων ὅσαι καθήκουσιν ἐπὶ τὴν θάλασσαν. ὧν εἰσὶν ἐν δεξιᾷ πρῶτοι Μαυρουσίων ὅσοι περὶ τὴν θάλασσαν, ὅσα τε ἄλλα Λιβύων ἔθνη μέχρι Καρχηδόνος, καὶ τούτων ὕπερθε Νομάδες, οὓς Ῥωμαῖοι καλοῦσι Νουμίδας καὶ τὴν χώραν Νουμιδίαν, ἕτεροι δὲ Λίβυες ὅσοι περιοικοῦσι τὰς Σύρτεις μέχρι Κυρήνης, Κυρήνη τε αὐτὴ καὶ Μαρμαρίδαι καὶ Ἀμμώνιοι καὶ οἱ τὴν Μάρειαν λίμνην κατοικοῦσι, καὶ ἡ μεγάλη πόλις ἣν Ἀλέξανδρος ἔθηκε πρὸ Αἰγύπτου, Αἴγυπτός τε αὐτὴ μέχρι Αἰθιόπων τῶν ἑῴων ἀνὰ τὸν Νεῖλον πλέοντι, καὶ μέχρι Πηλουσίου διὰ θαλάσσης.

2. Ἐπιστρέφοντι δὲ τὸν πλοῦν καὶ περιιόντι Συρία τε ἡ Παλαιστίνη, καὶ ὑπὲρ αὐτὴν

2

Nouns
air die Luft
amber der Bernstein
animal das Tier
apple das Apfel
arm der Arm
backpack der Rucksack
bag der Tasche
beach der Strand
bed das Bett
beef das Rindfleisch
beer das Bier
bike das Fahrrad
bill die Rechnung
gun die Pistole
hair das Haar
hand die Hand
hang bag die Handtasche
head der Kopf
hill der Berg
history der Geschichte
home das Zuhause
honey der Honig
horse das Pferd
house das Haus
jacket die Jacke
joke der Witz
juice der Saft

General Conversation

Arabic	*Pronunciation*	*Meaning*
السلام عليكم	as-salaam alaeikum	peace be with you (formal greeting)
كيف حالك؟	keifa haaluk	how are things? (formal)
ازيك	e-zayyak	what's up? (informal)
الحمد لله	al-hamdu lillah	praise God (formal answer)
أنا بخير	ana bi kheir	I'm good (formal answer)
تمام / كل تمام	tamaam / kullu tamaam	everything's good (informal answer)
قويس	q-wayyis	fine / pretty good (informal answer)
شو أخبار العمل؟	shoo akhbaar al-amal	how's work? (informal)
السلام عليكم	as-salaam alaeikum	peace be with you (formal goodbye)
مع السلامة	ma'as-salaama	bye (literally: with safety) (semi-formal)
بعدين	ba'dein	later (see you later) (informal)

Figure 1.2: Three categories of documents that are limiting the implementation of a robust OCR technology. Modern cursive scripts including Nastaleeq and Devanagari challenge present OCR methods with their complex grapheme structure. Historical documents are generally highly degraded and possess archaic orthographies, which are difficult to reliably OCR. Multilingual documents are difficult to OCR as they comprise of multiple grapheme structures and a wide variety of character labels.

1.2 Research Hypothesis

The main hypothesis of this research work is based on the claim that RNN very closely mimic the functioning of biological neural networks [GEBS04]. Therefore, they should be able to replicate human reading capabilities to a good extent. Their ability to process contextual information enables them to learn sequence to sequence mapping, making them an appropriate choice for text recognition tasks. Context plays an integral part in reliably predicting a character in this situation. Moreover, their ability to automatically learn the distinguishing features from a text-line image enables them to work on a variety of scripts and languages.

1.3 Contributions of the Thesis

The major contributions of this thesis in the field of OCR for modern and historical printed documents are summarized below:

1. A semi-automated database generation methodology for cursive scripts is presented. This methodology is evaluated on creating a ligature-level database for Urdu Nastaleeq script; however, the same method can be extended to other scripts to create character or word-level GT database.
2. Datasets for Polytonic Greek and printed Devanagari scripts are presented using the image synthesis methods that model various image defects to reflect the scanning artifacts. These databases are made freely available for other research purposes.
3. Application of LSTM-based methodology to OCR printed English, Devanagari, Urdu Nastaleeq, Fraktur, and Polytonic Greek scripts has been presented. This segmentation-free OCR approach yields the best reported results for these scripts on public databases.
4. A framework, combining segmentation-based and segmentation-free OCR approaches, is proposed to digitize medieval Latin script of the 15^{th} century. This framework can easily be adapted to other scripts where training data is not available. Starting with single characters, a segmentation-based OCR system is used to generate the text-line level, semi-corrected transcription. This semi-corrected ground-truth data is then used to train an LSTM-based segmentation-free OCR system in an iterative manner. With each iteration, the LSTM network learns more characters using the context of neighboring characters.

5. A sequence learning methodology has been presented to identify multiple scripts in a multilingual document. This method works at text-line level with high accuracy, yet it identifies the script of individual characters.
6. A novel approach is described to carry out language independent OCR for a single script that is used by multiple languages. LSTM-based OCR has shown that it can yield very low error rates to OCR the text in these languages using a single model trained on a mixture of texts.
7. The basic idea behind the language independent OCR (using only a single LSTM model) is extended to the more general case of script-independent generalized OCR where a single LSTM model can recognize text in multiple scripts.

1.4 Thesis Structure

The structure of this thesis is depicted in Figure 1.3 showing the relationship between different chapters. The first two chapters overview the field of OCR. The remaining thesis can be divided into three main parts. The first part, spanning Chapter 3 and Chapter 4, familiarizes readers with specific issues of modern and historical scripts, and states the contributions of this thesis in proposing various datasets for several scripts. The second part of this thesis, comprising of Chapter 5 and Chapter 6, reports the monolingual OCR systems developed for modern and historical scripts respectively. The third part consists of Chapter 7 and Chapter 8 and it discusses the contributions of this thesis for text recognition in multilingual documents. A chapter-wise overview of the thesis is now given:

The second chapter overviews the various methodologies that have emerged as a result of decades of OCR research. Two categories of OCR methods, one termed in the literature as *segmentation-based* and the other as *segmentation-free* techniques, are discussed. Both methodologies are benefited by many image processing, statistical pattern recognition, and machine learning algorithms. In particular, it is noted that modern machine learning algorithms, specifically recurrent neural networks are suitable to OCR text-line images in a sequence learning paradigm. An OCR method based on these networks has been utilized in this thesis for various scripts.

After discussing different methodologies for OCR, Chapter 3 outlines several challenges that are hampering the establishment of a reliable OCR system. Specific challenges due to the nature of various modern and historical scripts have been described. Moreover, it is also highlighted that the unavailability of GT data is a major obstacle to OCR reliably.

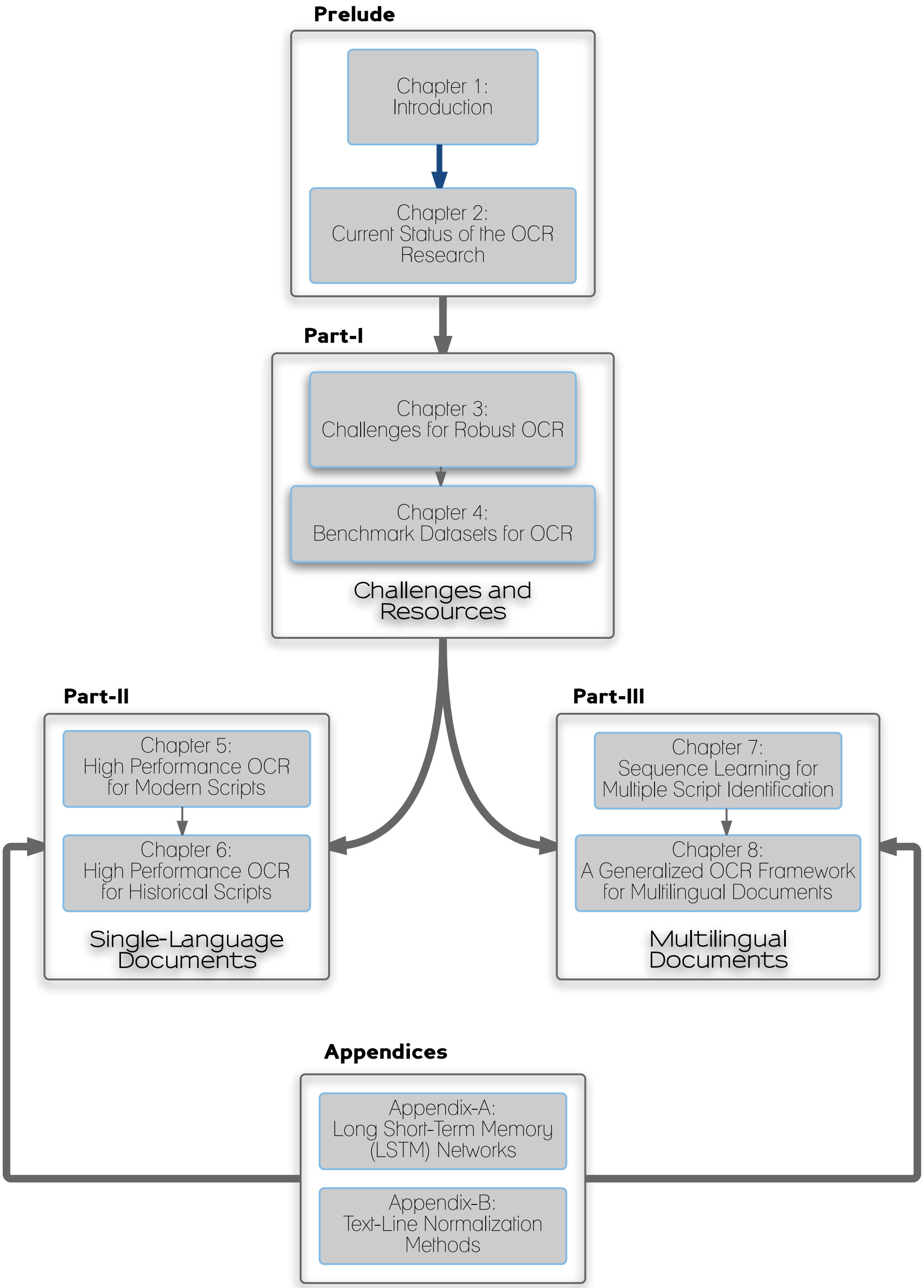

Prelude
Chapter 1: Introduction
Chapter 2: Current Status of the OCR Research
Part-I
Chapter 3: Challenges for Robust OCR
Chapter 4: Benchmark Datasets for OCR
Challenges and Resources
Part-II
Chapter 5: High Performance OCR for Modern Scripts
Chapter 6: High Performance OCR for Historical Scripts
Single-Language Documents
Part-III
Chapter 7: Sequence Learning for Multiple Script Identification
Chapter 8: A Generalized OCR Framework for Multilingual Documents
Multilingual Documents
Appendices
Appendix-A: Long Short-Term Memory (LSTM) Networks
Appendix-B: Text-Line Normalization Methods

Figure 1.3: Thesis structure and recommended flow of reading. Part-I should be read first after the prelude. Then Part-II and Part-III can be studied in parallel. The chapters within a part are inter-related, so it is highly recommended to read them in the shown sequence.

One issue that is of particular interest for many modern and historical scripts is the unavailability of datasets to evaluate the OCR algorithms. Chapter 4 presents the contributions of this thesis in the development of various datasets. These data collections are based on semi-automated database generation method and on synthetic text-generation using different image degradation models.

Chapter 5 discusses the results of OCR on three modern scripts using the above-mentioned LSTM networks. The LSTM-based OCR text-line recognizer yields excellent results without using any hand-crafted features or any sophisticated post-processing technique, like language modeling and dictionary correction.

Chapter 6 presents the evaluation of the same LSTM architecture for three historical scripts namely, German Fraktur, Polytonic Greek, and Latin script of medieval ages. It is demonstrated that using artificial data (generated using various image defect models) greatly helps in training an LSTM network that yields low error rates on these scripts. However, for Latin script of the medieval era, the transcribed data is not present either, making it extremely difficult to even produce synthesized data. The second part of this chapter presents the contribution of thesis in combining a segmentation-based OCR method (trained on individual characters) with an LSTM-based OCR approach.

Chapter 7 and Chapter 8 compose the last part of this thesis, which deals with the OCR of multilingual documents. Chapter 7 introduces a unique method for multiple script identification, using the sequence-to-sequence mapping principle of LSTM networks.

Script identification is traditionally used in the process of multilingual OCR to separate the underlying scripts before applying single OCR models on recognized scripts. However, script identification is unsatisfactory from the point of view of human reading process, which does not involve any such step. Based on this idea, a generalized framework to OCR multilingual documents is proposed in Chapter 8 that does not incorporate the script identification step. In the first part of this Chapter, a language independent OCR method is proposed using LSTM networks. A single OCR model trained on a mixture of three European languages of the Latin script yields very low error rates when tested on each of the individual language. An extension of this work is reported in the second part of this chapter to include multiple scripts.

Chapter 9 concludes this thesis with a description of some important lessons learnt and provides some guidelines for future work.

Chapter 2

Research Methodologies for Printed OCR

OCR is an old discipline, dating back to 1870 when the first retina scanner was invented by C. R. Carey. The first patent on OCR (named "Reading Machine") was obtained in 1929 [Tau29]. The first commercial OCR system was made available in the 1950s to digitize sales reports [Eik93]. The OCR machines available till the 1980s were primarily hard wired systems with specialized fonts and styles. These earlier systems were very costly, thereby restricting their common use in public. However, as computer hardware became cheaper, the growth and demand of such equipment started to increase. By the mid 1980s, software-based OCR systems became available. Modern OCR systems have combined the benefits of both advancement in technology and hardware. This chapter overviews some of the broad categories that have emerged over a century of research in this field.

Traditionally, the process of recognizing text involved the segmentation of the page into text-zones (paragraphs), then into text-lines, followed by words and finally into individual characters. A character recognizer would recognize the segmented characters one by one. This technique is referred to as "segmentation-based" OCR in the literature and Section 2.1 describes various methods developed for this type of OCR over the years.

The second, more recent technique is holistic OCR. In this approach, the recognition is done at word or at text-line level. This avoids the harder task of extracting characters from text-lines. Section 2.2 discusses the current state of research in this category, in detail.

2.1 Segmentation-Based OCR

In segmentation-based OCR approach, the first major step is to identify individual components, suitable for final recognition (OCR) step. Such an approach depends heavily upon the extraction of individual characters from the text. This methodology has remained state-of-the-art for a very long time, before finally giving way to segmentation-free approaches. Tesseract [Tes14] – a popular open-source OCR system – is a fine example of a segmentation-based OCR system. Within this paradigm, there are many methods for both printed and handwritten character segmentation and various survey papers have been reported in the literature [CL96, Lu95, SSR10] for these methods.

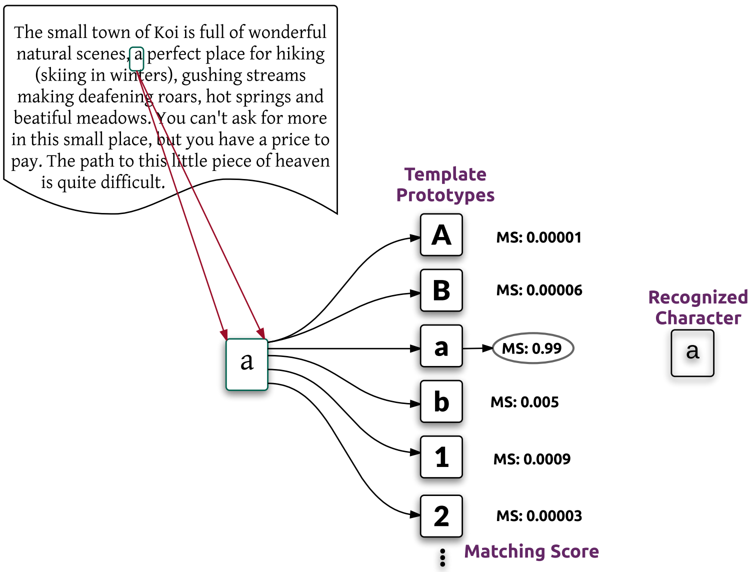
Segmentation-based OCR can be broadly divided into two major classes. The first class is based on ***template matching***, in which connected components (characters) are extracted and are subsequently matched against the possible templates. The recognition is achieved based on a certain similarity criterion. Section 2.1.1 describes various ways in which this technique has been used for the purpose of OCR.

The second type of segmentation is based on ***over-segmentation*** technique. In this technique, instead of finding an exact dividing point between two characters, an approximate segmentation point (cut) is found. The over-segmentation is then corrected at a later stage. More details can be found in Section 2.1.2.

2.1.1 Template Matching for OCR

Tauscheck's [Tau29] patent of OCR was based on template matching. However, unlike modern day template matching algorithm for pattern recognition, templates were mechanical. A match is found when light in optoelectronic sensors fails to reach the detector. In fact, many early OCR systems were based on the mechanical template matching principle similar to the mentioned above [Han62, ERA57, IOK63].

As the computer technology became available, the template matching algorithms started appearing as a software-based solution. In a software-based template matching process for OCR, individual characters are extracted and then are matched against all possible templates of the characters (see Figure 2.1 for an illustration of this process). These templates are the representative prototypes of each character class. The actual matching is done pixel-by-pixel and the match is found when number of



The small town of Koi is full of wonderful natural scenes, a perfect place for hiking (skiing in winters), gushing streams making deafening roars, hot springs and beatiful meadows. You can't ask for more in this small place, but you have a price to pay. The path to this little piece of heaven is quite difficult.

Figure 2.1: Illustration of basic template matching. In template matching, an individual component (a character) is matched against different shapes' prototypes by simply superimposing the prototype on top of the character. Individual character's correspondence determine the similarity score.

matched bits (pixels) are higher than a certain threshold. The basic similarity measure used in practice is a *cross-correlation* function, given by the following equation.

$$X(x,y) = \frac{\Sigma_{x,y}\,[I(x,y)-\bar{I}_{u,v}][T(x-u,y-v)-\bar{T}]}{\sqrt{\Sigma_{x,y}[I(x,y)-\bar{I}_{u,v}]^2\Sigma_{x,y}[T(x-u,y-v)-\bar{T}]^2}}$$

where

- I is the image and T is the template
- $\bar{T}$ is the mean of the template
- $\bar{I}_{u,v}$ is the mean of $I(x,y)$ and is the region under the template

The above-mentioned process is closely related to the process of using a *convolution mask*[1], that is, the template of a specific character.

In addition to matching images directly, feature-based template matching is also well-known. In this methods, features of templates (prototypes) are first stored instead of the images themselves. For a query image, same features are extracted, which are then matched with the representative prototype's features.

---

[1] the cross-correlation of two functions f(x) and g(x) is equal to convolution of f(x) and g(-x)

In practice, the use of template matching based methods is quite limited as this method greatly suffers from image noise, font variations, touching characters, etc. Therefore, more sophisticated methods have been devised to either segment the characters more efficiently, or to use integrated segment-recognize methods, such as HMMs or RNNs with HMM-like input-output alignment methods.

2.1.2 Over Segmentation Methods

Over segmentation approach is applicable to situations where correct character segmentation is not possible. This may happen due to the presence of touching characters in a document. Instead of perfectly segmenting characters, *character candidates* are found by imperfect segmentation (see Figure 2.2). These character candidates are then recognized by standard pattern recognition approaches including nearest neighbor classifiers, decision trees, Bayes classifiers or neural networks. A simple heuristic is used to discard those candidates for which the confidence score of the classifier is below a certain threshold. However, using heuristics to find the right character candidate is not very useful in practice. There could be many artifacts like broken characters, fonts, etc. that may result in over-segmenting certain characters, for instance, ‘m’ can be segmented into ‘nn’ or ‘rn’ or ‘d’ can be confused with ‘cl’. A better way to recognize these character candidates is to use a language model to decide the correct hypothesis among all possible character candidates. During the language modeling phase, costs of appearing in certain combinations of character(s) are used to determine the correct choice.

There are many methods used to determine the approximate boundary between two characters [MWW13]. The segmentation point between two neighboring characters can be found by (i) *projection-based* methods, where the segmenting point is found by a vertical project profile, (ii) *feature-based* methods, where certain features are used to identify the character’s approximate boundary, or (iii) *skeleton analysis-based* methods, where a character’s thinned version is used to determine the dividing point.

2.2 Segmentation-Free OCR

Nagy in his famous work [Nag92] urged the document analysis community to move from the isolated character recognition to more holistic approaches that can incorporate contextual information in the recognition process. As a result, segmentation-free OCR approaches appeared, which recognize a single character on the basis of its

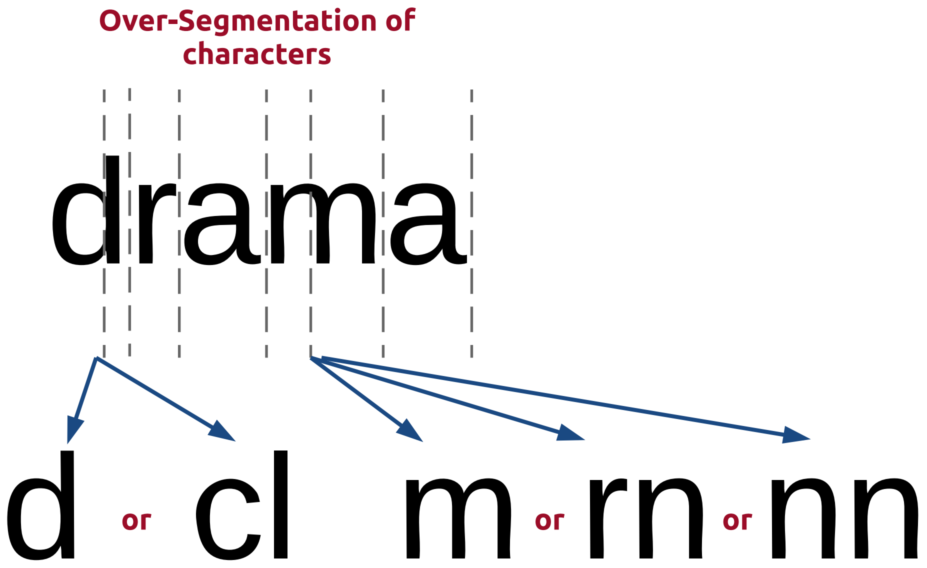


Figure 2.2: Illustration of segmentation process using over-segmentation method. The approximate segmentation points are found and each character candidate is recognized with a confidence score. The cost of occurring a character after the other is estimated using language model. The character candidate with lowest cost is selected as the recognized character.

surrounding context in addition to its own features. Segmentation-free approaches for OCR, differ from the segmentation-based methods in the level of segmentation required from layout analysis step. Traditionally such approaches need the extraction of words or part-of-words for a given text-line. The whole word or part-of-word are then recognized by the recognition engine. In holistic approaches, discriminating features are extracted from the whole word, part-of-words, or ligatures. These features are then used to train a classifier (HMM or ANN) to recognize the whole word or complete ligature. Segmentation-free approach is also referred to as integrated segmentation/recognition or implicit recognition methodology in the literature. These approaches perform the character localization and recognition at a text-line level; thereby avoiding the need of explicit word or character segmentation.

A very well-known segmentation-free approach is based on HMMs, which has remained the most popular OCR method for a long time. OCR methodology using HMMs is further described in Section 2.2.1. A more recent approach in segmentation-free domain is to use RNN in combination with CTC approach. More details on this approach are detailed in Section 2.2.2.

2.2.1 HMM-Based OCR Approach

Hidden Markov Models (HMM) are stochastic models in which the underlying process is assumed to be governed by the Markov Property[2]. For segmentation-free OCR, each character, word or ligature is modeled by a fixed number of states as shown in Figure 2.3. A fix-sized window traverses over the text-line. Features are extracted from this window and their vector is associated with states through a separate probability, called *emission probability*. During the training, the *state transition* probabilities are learned for each character, word or ligature in a text-line, and are represented as a *transition matrix*. A forward-backward algorithm [Rab89] aligns the output labels with the ground-truth data.

HMM Toolkit [YY94] is a popular choice among researchers to apply HMM for various tasks. Further details regarding the mathematics behind the HMM methodology and various underlying algorithms can be found in [Rab89]. HMM-based segmentation-free approach remains the dominated methodology for speech processing and handwriting recognition for over a decade. They have also been utilized for printed OCR, a few examples of which can be seen in [MSLB98, EM08, HHK08].

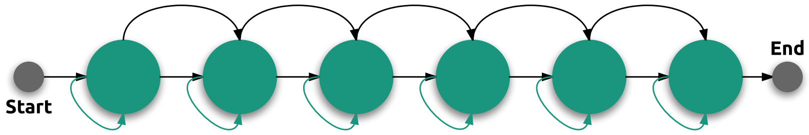


Figure 2.3: Figure showing the basic unit in HMM-based OCR. Each character is represented by a fixed number of states. The state transition probabilities (going from one character to the other) is learned using the HMM forward-backward algorithm.

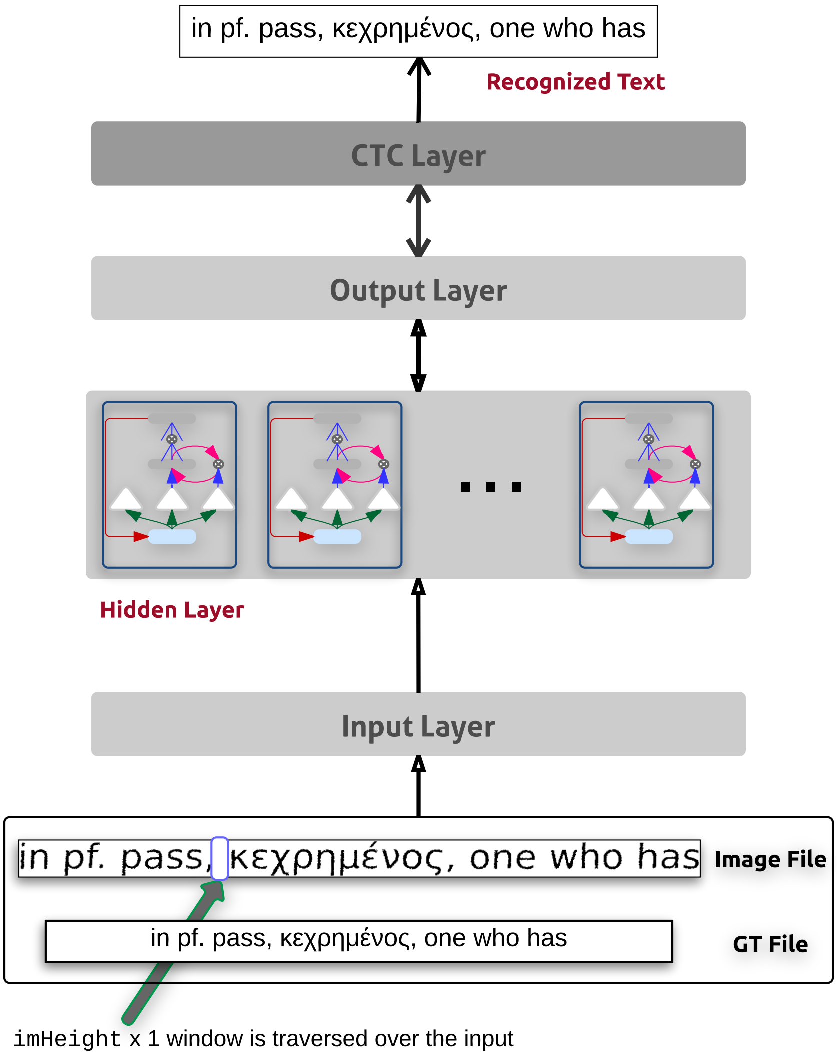
2.2.2 Sequence Learning Approach

In sequence learning (or sequence labeling as denoted in [Gra12]), the supervised learning is carried out on full sequences instead of individual components. Both input and targets are in the form of a sequence and the job of a classifier is to do sequence-to-sequence mapping. RNNs are used as core classifiers in contemporary sequence learning tasks. The feedback connections between the neurons at the hidden layers in such networks allow them to remember the context of a sequence; thereby making them learn the long contextual information. However, in practice they have not demonstrated the power of remembering very long context. The main problem lies in the use of gradient descent based learning where error signal (the difference between the actual output and the target output) is propagated back to update the internal weight connections. It has been shown [HS97, BSF94] that the values of first order gradient values either grow exponentially, *Exploding Gradient Problem*, or they vanish to zero exponentially, *Vanishing Gradient Problem*. These problems make the RNN learning very slow and impractical. The problem of training an RNN has been solved by Hochreiter and Schmidhuber [HS97] by replacing the traditional activation units at the hidden layer with a memory cell, termed as Long Short-Term Memory (LSTM), with capability to retain the gradient values for longer times.

---

[2] the current state of the process is independent of its previous states

However, neural networks require segmented input, so that they can learn the association of that input with the target class. This requirement of input segmentation renders them unsuitable to learn the complete sequences. In LSTM-based sequence learning tasks, a specialized algorithm, described in literature as the *Connectionist Temporal Classification (CTC)*, which is similar to HMM's forward-backward algorithm, is used to align the output activations of a neural network with target labels. The output of such learning algorithms is a sequence of discrete labels. For text recognition tasks, these labels are the characters of a given family of script. Figure 2.4 shows an example of this using sequence learning paradigm.


in pf. pass, κεχρημένος, one who has
Recognized Text
CTC Layer
Output Layer
Hidden Layer
Input Layer
in pf. pass, κεχρημένος, one who has
Image File
in pf. pass, κεχρημένος, one who has
GT File

`imHeight` x 1 window is traversed over the input image to convert the 2D image into 1D sequence

Figure 2.4: Illustration of how RNNs are used for the OCR task. In sequence learning paradigm, the RNN is trained in a supervised manner using full sequences (text-line image) and the corresponding ground-truth sequence (the ground-truth text-line). The output activations are aligned with the ground-truth using a forward-backward algorithm (CTC).

Chapter 3

Challenges for Robust OCR

Baird and Tambore have outlined many *stubborn obstacles* to document image recognition in [BT14] that are hindering the realization of a robust OCR system. This chapter focuses on two specific challenges among them: 1) challenges that occur due to the nature of the script, and 2) those that arise due to the unavailability of Ground-Truth (GT) data. The nature of a script entails its graphemes and the associated typographic rules needed to join them. The grapheme structure, also referred by some researchers as *morphology of a script* [BR14], directly affects the performance of an OCR system. Complex scripts are thus inherently more problematic for OCR algorithms. Likewise, typographic rules in some scripts lead to the formation of ligatures and compound shapes, which further complicate reliable recognition of such scripts.

Moreover, historical documents pose additional issues to those offered by modern scripts. The aging of a document can severely effect the text in many ways, such as, deformation of text-lines, low contrast, missing information due to page tear, page thinning, bleed through, shine through, and ink spread. Furthermore, multilingual documents (both modern and old) also give rise to the complexity because of the presence of multiple orthographies and scripts.

The second challenge in the development of reliable OCR is the unavailability of Ground-Truth (GT) or transcribed data. The training of OCR models using the supervised Machine Learning (ML) requires large amounts of data. The diversity of documents makes it very hard to capture the complete representation of data in training OCR models [BT14]. The lack of GT thus greatly hampers the development of OCR systems for many languages and scripts.

The first part of this chapter, from Section 3.1 to Section 3.4, discusses the difficulties faced in the OCR of Latin and Polytonic Greek scripts. These scripts possess simple grapheme structure from the perspective of OCR; however, there are further OCR challenges, such as, similar shaped characters, decorative styles, and the presence of diacritics, which make the OCR difficult. The second part, which includes Section 3.5 and Section 3.6, explains the difficulties faced in OCR of South Asian scripts (Devanagari and Urdu Nastaleeq), due to the complex grapheme structure and complicated typographic rules. The final part, Section 3.7, describes the OCR challenges posed by various types of multilingual documents. Throughout the current chapter, the lack of appropriate databases is highlighted as one of the major issues in developing dependable OCR systems.

3.1 Modern English

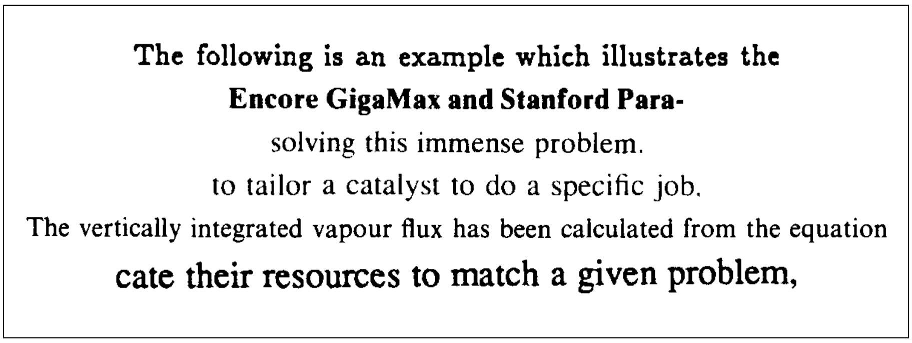
Over decades, text written in modern English has remained the main focus of OCR research. Latin script, or sometimes referred to as the *Roman* script, is used to write the English language. At present, the OCR for printed Latin script is considered to be a solved issue by many researchers. The grapheme structure of Latin script is comparatively simpler than oriental scripts. However, research in printed English OCR (Latin script) is still pursued to date. Some of the unresolved issues that are keeping the research in this area alive are highlighted in the following sub-section.

3.1.1 Challenges for OCR in Printed English

Similarity of Shapes: There are many similar-shaped characters in modern English typography. Such characters or part of characters make it difficult for the OCR engine to recognize them reliably. Some of these characters are ‘l’ and ‘1’ or ‘i’ and ‘l’, ‘cl’ and ‘d’, ‘rn’ and ‘m’. These issues are particularly more prominent in segmentation-based OCR systems (see Section 2.1).

Similar Capital and Small Characters: Some characters have the same shape in both of their capital and small form, e.g., ‘o/O’, ‘s/S’, ‘x/X’, and ‘v/V’. These character pairs confuse the shape-matching algorithm for correct recognition and thus results in poor recognition accuracy.

Degradations in the Documents: Introduction of noise in scanned documents is a common phenomenon [NS14]. It can distort the characters to a level where they are no longer recognizable by an OCR algorithm. The usual process of many



The following is an example which illustrates the
Encore GigaMax and Stanford Para-
solving this immense problem.
to tailor a catalyst to do a specific job.
The vertically integrated vapour flux has been calculated from the equation
cate their resources to match a given problem,

Figure 3.1: Sample text-line images from UW-III database. UW3 database is the standard dataset for Latin OCR algorithms. In this thesis, a subset of this database excluding equations and tables has been used.

DIA systems is to scan the page and apply binarization before further processing. There are multitude of artifacts that appear in documents during the capturing process, for example, ink spread, bleed-through (where part of the text appears on the other side of page), skew, and salt-and-pepper noise. Some of these degradations are corrected at the binarization stage, which employs various heuristics to remove these artifacts. A downside of this process is that many characters get distorted even further and other characters get joined together. This results in poor segmentation and increased recognition errors.

Camera-Captured Documents: Due to the availability of very cheap capturing devices, there is an increase in the documents captured using hand-held cameras. Such documents pose several unique challenges, like text image with curled text-lines, motion blur and out-of-focus text images. The standard algorithms for OCR do not work well on these documents.

Text Recognition in Natural Scene and Videos: Another related issue in reliable text recognition is the text in natural scenes and videos. The recognition of text in such a scenario requires text localization (where the text is located in the image) and text segmentation (separating text from the background).

3.1.2 OCR Databases for modern English

There are plenty of databases available for printed English script to check the OCR algorithms as a result of continued research in this field for many decades. Two very popular data sets, which have been used in this thesis, are described below.

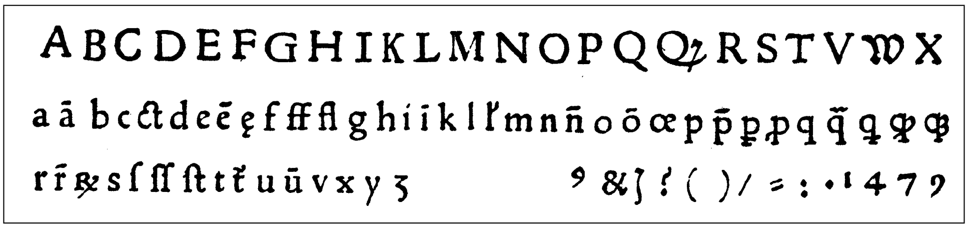


Figure 3.2: Figure showing the character set used in the old Latin documents. Many characters are quite similar and a lot of them are no longer in use today.

University of Washington (UW3) Text Image Database: The text image database, UW3 [LRHP97], consists of 1600 document images with GT information for document layout and text recognition tasks. GT is available for equations, tables and captions in addition to the normal text. For the purpose of our research, only normal text-lines have been considered. Some of the text-lines from this database are shown in Figure 3.1.

UNLV-ISRI Database: UNLV-ISRI database [TNBC00] consists of 1056 documents with 46,586 pages. Similar to the UW3 database, only normal text-lines have been used from this database.

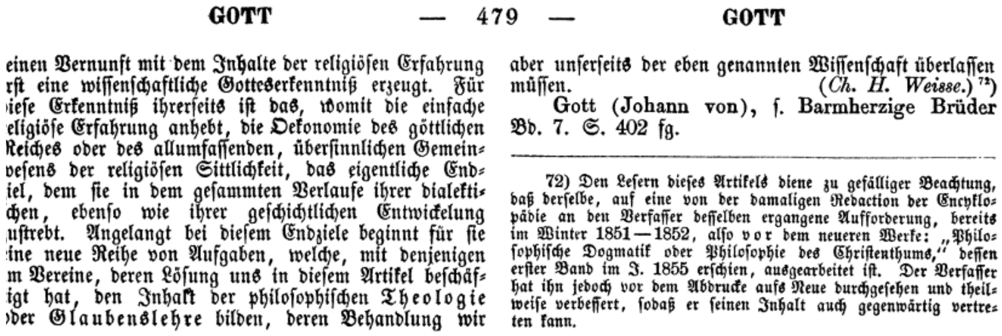
3.2 Latin Script of Late Medieval Ages

Medieval Latin script contains a variety of characters that are not found today in the modern Latin script (see Figure 3.2 for an example). A wealth of literature is available in the old script and there is a high requirement for digitizing this literature to preserve the literary heritage.

3.2.1 Challenges for Old Latin OCR

There are many aspects of historical documents that directly or indirectly affect the reliability of OCR. Some of these issues are highlighted below.

Aging Effects: Aging severely affects the paper quality of historical documents. Unsuitable ambient environments cause many artifacts, for example, curled text-lines, page tear, pages sticking together, faint typing, and page thinning. The page thinning results in bleed-through, ink-spread, color blotting, etc.



GOTT — 479 — GOTT

einen Vernunft mit dem Inhalte der religiöſen Erfahrung
rſt eine wiſſenſchaftliche Gotteserkenntniß erzeugt. Für
ieſe Erkenntniß ihrerſeits iſt das, womit die einfache
eligiöſe Erfahrung anhebt, die Oekonomie des göttlichen
Reiches oder des allumfaſſenden, überſinnlichen Gemein=
veſens der religiöſen Sittlichkeit, das eigentliche End=
iel, dem ſie in dem geſammten Verlaufe ihrer dialekti=
chen, ebenſo wie ihrer geſchichtlichen Entwickelung
uſtrebt. Angelangt bei dieſem Endziele beginnt für ſie
ine neue Reihe von Aufgaben, welche, mit denjenigen
m Vereine, deren Löſung uns in dieſem Artikel beſchäf=
igt hat, den Inhalt der philoſophiſchen Theologie
der Glaubenslehre bilden, deren Behandlung wir

aber unſerſeits der eben genannten Wiſſenſchaft überlaſſen müſſen. (*Ch. H. Weisse.*)[72])

Gott (Johann von), ſ. Barmherzige Brüder Bd. 7. S. 402 fg.

72) Den Leſern dieſes Artikels diene zu gefälliger Beachtung, daß derſelbe, auf eine von der damaligen Redaction der Encyklopädie an den Verfaſſer deſſelben ergangene Aufforderung, bereits im Winter 1851—1852, alſo vor dem neueren Werke: „Philoſophiſche Dogmatik oder Philoſophie des Chriſtenthums,“ deſſen erſter Band im J. 1855 erſchien, ausgearbeitet iſt. Der Verfaſſer hat ihn jedoch vor dem Abdrucke aufs Neue durchgeſehen und theilweiſe verbeſſert, ſodaß er ſeinen Inhalt auch gegenwärtig vertreten kann.

Figure 3.3: Example of Fraktur script. Ersch-Gruber is an encyclopedia written in the Fraktur script.

Archaic Orthography: Like many other historical scripts, old Latin also exhibits a vast variety of characters that are no longer in use today. Figure 3.2 shows the complete character-set of Latin script of 15th century. Many characters are similar to each other in shape and a lot of them are no longer in use.

3.2.2 OCR Databases for Old Latin

The major issue faced by researchers in digitizing old documents is the lack of GT data for training and evaluation of OCR algorithms. Using synthetically generated data could be a solution, but to generate it, one needs to have already transcribed data. Transcribing these documents is in itself a challenging task and requires script-specific expertise; thereby rendering it very expensive. The IMPACT project[1] aimed at digitizing large collections of historical text in various European languages. Many new tools and methods have emerged as an outcome of this project, but many challenging historical scripts still require further research.

3.3 German Fraktur Script

Fraktur[2] remained a very popular script in Central Europe, especially in Germany between 16th century and the middle of the 20th century. This script is still being used, especially in Germany, by many newspapers for their titles. Figure 3.3 shows a scanned page from Ersch-Gruber Encyclopedia, which was written in the Fraktur script.

---

[1] http://www.dlib.org/dlib/january09/01inbrief.html#PLOEGER

[2] https://en.wikipedia.org/wiki/Fraktur

3.3.1 Challenges for Fraktur OCR

Decorative Style: The decorative nature of this script makes it difficult for OCR algorithms to work correctly. Moreover, the shape of some letters resemble very closely to that of others (see Figure 3.4 for such shapes).

𝖚 𝖓 (u n), 𝕭 𝖁 (B V), 𝕽 𝕹 𝕶 (R N K),
𝕴 𝕵 (I J), 𝕬 𝖀 (A U), 𝖉𝖇 (d b), 𝖝 𝖗 (x r)

Figure 3.4: Shape confusion in Fraktur script. Many characters in Fraktur resemble very closely to other characters. Characters in parentheses show the same characters in modern Latin typesetting.

Degraded Documents: Another issue regarding old documents in general is their scan quality. Many a times, they are not in a good condition, so when one applies the preprocessing steps such as document binarization, some characters lose their structure, such as ‘u’ and ‘n’ as shown in Figure 3.5.

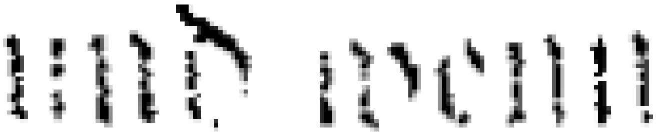


Figure 3.5: Document quality degradation caused during preprocessing. Binarization may result in broken characters which will then be a challenge to recognize. The actual words are ‘und wenn’.

3.3.2 OCR Databases for Fraktur

Lack of a standard database to establish OCR methods for Fraktur script makes it is quite hard to compare the OCR error rates for various algorithms. Large amount of transcribed data is required to train the supervised learning based OCR systems. Preparing the data manually for this task is laborious and tedious. This becomes even more complex in the case of historical documents due to the presence of ancient orthographies and scripts. Language-specific experts are required to transcribe a given historical document, which makes the manual transcription extremely costly. Chapter 4 and Chapter 6 of this thesis discuss various ways of addressing the challenge of limited data.

3.4 Polytonic Greek Script

Polytonic orthography has remained the dominant form of Greek writing in both ancient and Medieval ages till the emergence of monotonic orthography (modern Greek) in 1982. This section describes some challenges posed by this historical script in digitization of Greek literary heritage.

3.4.1 OCR Challenges for Polytonic Greek

The appearance of various diacritics in Greek orthography in 3^{rd} century BC produced many groups of different symbols for each vowel character that looked similar. These diacritics included, (i) the acute accent **oxeia** – *sharp or high*, (ii) the grave accent (**bareia** – *heavy or low*), (iii) the circumflex (**perispomene** – *twisted around*), (iv) the rough breathing **dasi pneuma**, (v) the smooth breathing **psilon pneuma**, (vi) the **diaeresis** to indicate diphthong, and (vii) the **iota** subscript (hypogegrammene *written under*).

These diacritics and their combinations can be associated with the 14 vowel characters (7 upper-case and 7 lower-case letters) according to the various phonological and orthographical rules developed in different periods of Greek history. For example, the Greek letter *α* may have the following variations: *αάἀἁἂἃἄἅἆἇὰᾶᾰᾱᾳᾀᾁᾂᾃᾄᾅᾆᾇᾲᾷᾴ*. The groups of very similar characters result in a large character-set (more than 200), and they make the OCR of this script very challenging.

3.4.2 OCR Databases for Polytonic Greek

Lack of GT data is hindering the development of Polytonic Greek OCR systems. A large collection of transcribed data is collected under a Greek government funded project called OldDocPro [GSL$^+$][3] to recognize machine-printed and handwritten polytonic documents.

Up till now, the focus of this chapter has been to identify OCR challenges in those scripts whose grapheme structures are simple. In the next two sections, the challenges raised by two South-Asian scripts, namely Devanagari and Urdu Nastaleeq are explored, whose grapheme structures are more complex than Latin or Greek scripts.

---

[3] `http://www.iit.demokritos.gr/~nstam/GRPOLY-DB`

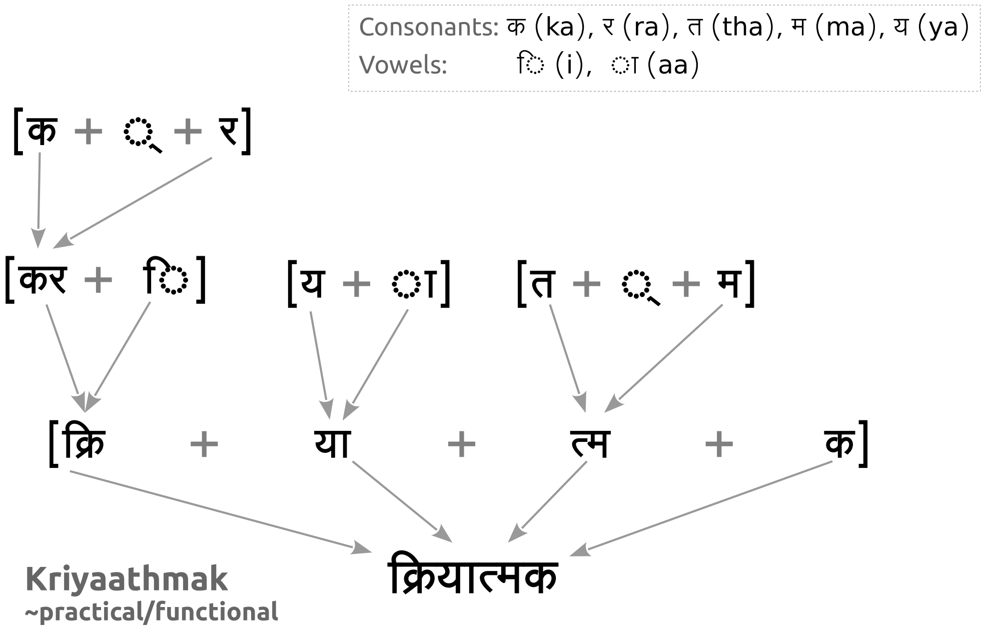


Figure 3.6: Word formation in Devanagari script. Word formation in Devanagari script. Note that how vowel ि (i) joins before the conjunct and how "र" (ra) changes shape when joined with a consonant (first row).

3.5 Devanagari Script

Devanagari belongs to the Brahmic family of scripts, which are popularly found across the south eastern part of Asia. Languages like Sanskrit, Hindi and Nepali are few of the popular languages that use the Devanagari script. A great wealth of ancient classical literature, scientific and religious books are available in Hindi/Sanskrit. Therefore, there is an increasing requirement to convert them into machine readable documents.

Devanagari is a cursive script, where words are formed by joining consonants and vowels; however, the shapes of individual vowels and consonants change while forming a word. The process of writing a word in Hindi is illustrated in Figure 3.6. The necessary vocabulary is given on the top-right side. Two consonants are first joined through the special character ् (a dashed circle with a diagonal dash beneath) that is used to remove the inherent vowel of the first consonant. At the second stage, this conjunct is combined with the vowel ि (pronounced as 'ee'). This vowel when combined with the conjunct appears at the beginning of the composite glyph. Similarly, the consonant य (ya) is combined with vowel ा (aa) to make या (yaa), and the consonants त (th) and म (ma) are combined to make त्म (thma). The last letter क (ka) is then attached to the word and whole thing is combined with a horizontal line to make the word *kriyaathmak*, meaning *practical* or *functional*.

3.5.1 Challenges for Devanagari OCR

Even though there is a great demand for Devanagari OCR, research in this field is still behind that of the Latin script. Devanagari script is much more complex as compared to Latin or Greek. The character shapes in Devanagari are far too many and vary widely with different fonts. Each of the consonant-consonant conjuncts and consonant-vowel combinations take different forms based on their position in the word. Some specific challenges are outlined below:

Consonant-Vowel Combination: Consonants when attached to vowels, other than the inherent vowel 'a', are modified with diacritics or *Maatra* (as they are called in Hindi). Diacritics or Maatra are glyphs when added to the letter, modify its shape. For example, the consonant त (tha) when combined with the vowel इ (e) will take the form ति (thi). The glyph 'ि' , which is added to त , is called as a *Maatra*.

Consonant-Consonant Combination: Consonants within a word also fuse together to form new shapes. For example, consonants त (th) and म (ma) when present in a word, fuse together to form त्म (thma). These are called *Samyuktakshara*, meaning 'conjuncts'.

The above-mentioned shape variations and consonant-vowel or consonant-consonant connections make OCR for Devanagari a difficult task.

3.5.2 OCR Databases for Devanagari

Apart from the complexity of the language, the lack of standard databases is another challenge faced by researchers in proposing a reliable OCR system for Devanagari. There are few databases used in various works on Devanagari OCR reported in the literature, but most of them are private databases, used by the researchers for certain experiments and they are either no more in use or are not available anymore.

3.6 Urdu Nastaleeq Script

Like Devanagari, Urdu Nastaleeq script also possesses a complex structure. This script is the predominant script currently being used to write Urdu and Persian Languages.

It was developed in the 8^{th} and the 9^{th} centuries in Iran. Since then, it has been extensively used in Iran, Pakistan and Afghanistan till today to write daily newspapers, books, and other literature. At this point, it is important to distinguish the Nastaleeq script from *Naskh* script, which is the primary script to write Arabic. Urdu Nastaleeq belongs to the family of Right-to-Left (RTL) languages that are written and read from right-to-left. However, in Urdu, the numbers are read from left-to-right. Figure 3.7 shows an Urdu text-line where numbers should be read similar to English. The remaining text is written and read from right-to-left. Digital content of present-day Urdu is also developed using this script.

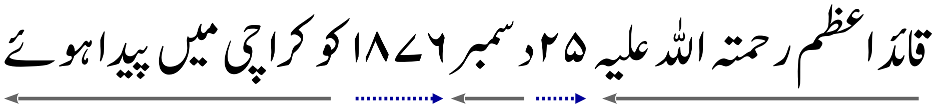

قائداعظم رحمتہ اللہ علیہ ۲۵ دسمبر ۱۸۷۶ کو کراچی میں پیدا ہوئے

Figure 3.7: Reading direction in Nastaleeq script.Nastaleeq script is read from right-to-left. However, numbers are to be read from left to right. Solid arrows point in the reading direction of normal text, while the dotted arrows point in the reading direction of numbers.

3.6.1 Challenges for Nastaleeq OCR

Urdu Nastaleeq offers unique challenges to the task of OCR. In the following discussion, we will discuss the main properties of Urdu Nastaleeq script that are curtailing the realization of a reliable OCR system for this script.

Diagonal Flow of Writing: To use the paper more efficiently[4], this script was developed to flow diagonally, from right to left. When a character is attached to another joinable character, the preceding character shifts upwards to the right, thus making the flow diagonal from right-to-left (See Figure 3.8). This arrangement makes a word to occupy less space.

On the other hand, although the joining rules for Naskh script are the same, the writing flow remains horizontal from right-to-left. Figure 3.9 shows the same text-line in both scripts.

From the perspective of OCR, the Nastaleeq script is more challenging than the Naskh script because it is very hard to distinguish some characters in a word or sub-word in the former one, where neighboring characters overlap each other in a diagonal flow.

---

[4] paper was a rare commodity at the time when this script was being developed

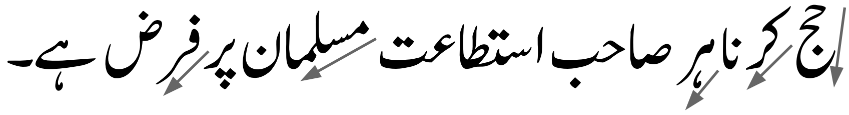
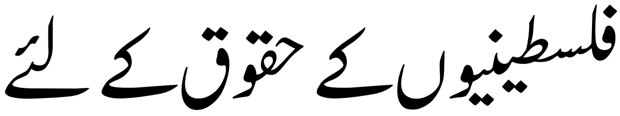
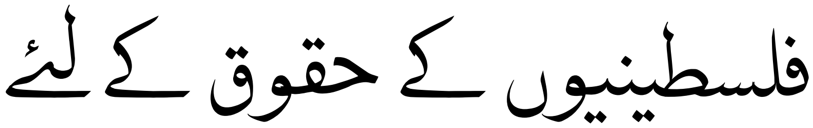

حج کرنا ہر صاحب استطاعت مسلمان پر فرض ہے۔

Figure 3.8: Diagonal writing flow in Nastaleeq. Nastaleeq script flows diagonally from right top to left bottom. This arrangement condenses the word shape and makes it fit in a small space.


فلسطینیوں کے حقوق کے لئے

Nastaleeq Script


فلسطینیوں کے حقوق کے لئے

Naskh Script

Figure 3.9: Comparison between Nastaleeq and Naskh scripts. The same text utilizes more space while written in the Naskh script compared to Nastaleeq where it takes up less space. This is due to the fact that the Naskh script flows horizontally from right-to-left, while the Nastaleeq script flows diagonally from right top to left bottom.

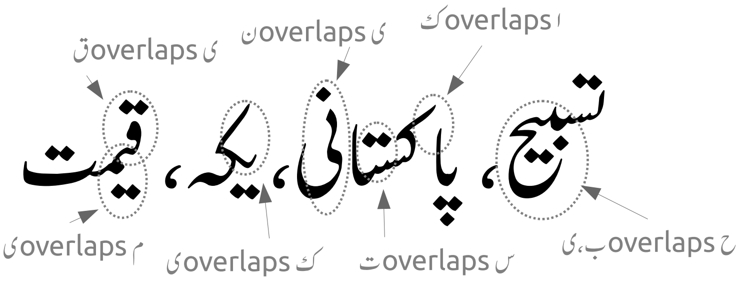

ق overlaps ی
ن overlaps ی
ک overlaps ا
تسبیح، پاکستانی، یکہ، قیمت
ی overlaps م
ی overlaps ک
ت overlaps س
ب،ی overlaps ح

Figure 3.10: The issue of characters overlapping within a ligature or between two adjacent ligatures.

Character Overlapping: Due to the diagonally flowing structure, many characters overlap the neighboring characters. This makes the word segmentation considerably harder in Nastaleeq script as compared to Latin or Naskh scripts. Figure 3.10 shows a text-line highlighting this issue.

Character's Shape Change: A common characteristic of both Naskh and Nastaleeq script is that the shape of a character changes depending on its position in a

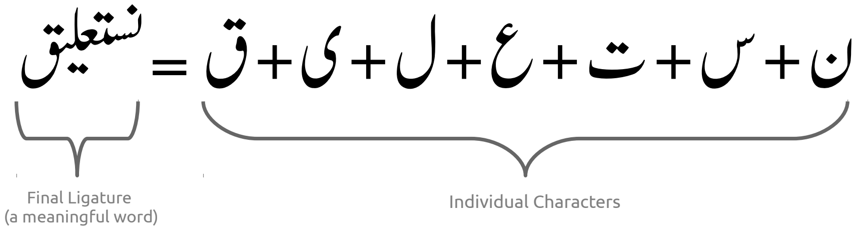


Figure 3.11: A word in Urdu with its constituent characters.

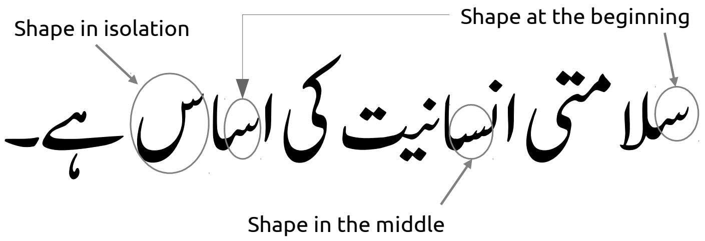


Figure 3.12: The shape of a character at different positions in a word. A single character can take different shapes when it appears at different positions.

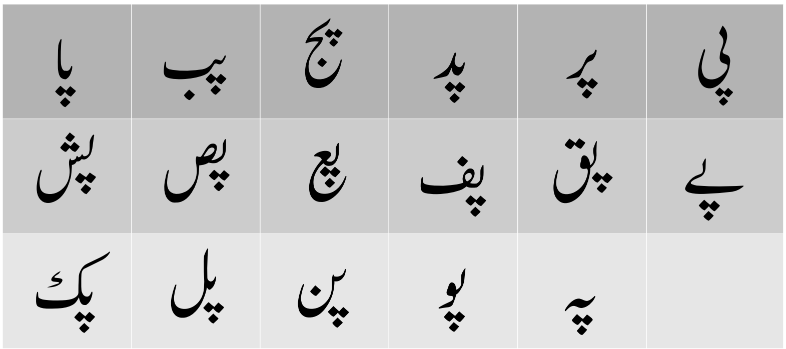
word or ligature[5]. A character can occur in four scenarios in a word: in isolation, at the beginning, in the middle, or at the end of a word or ligature. Figure 3.11 shows a word and its constituent characters. Note how the shape of individual characters change when they combine with others. Figure 3.12 shows the shape variation of a single character when appearing at different positions in a word.

Contextual Shape Change: In Nastaleeq script, the shape of a character does not necessarily remain the same in a single position. Unlike Naskh, where the shape of a character in a certain position is independent of the preceding or succeeding character, the character shape in Nastaleeq at a single position depends upon the preceding and succeeding characters. Therefore, there are many shapes possible for a single character in a specific position. Figure 3.13 shows an example of letter پ (*pay*) appearing in three joinable positions (beginning, end, and in the middle) with different characters.

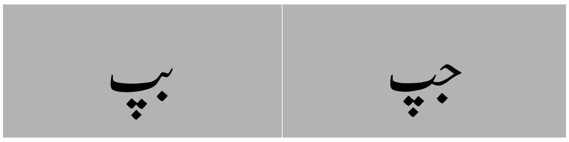
No Concept of Baseline and x-height: The x-height (the height of letter ‘x’ in Latin) and baseline are considered important distinguishing features for many Latin characters [BUHAAS13] for OCR. However, there is no concept of baseline and x-height in the Nastaleeq script. Due to the diagonal nature of the script, the

---

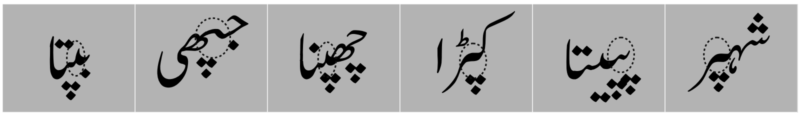
[5] Ligatures are defined as a single glyph consisting of two or more graphemes.



Shape change when joining to a succeeding character

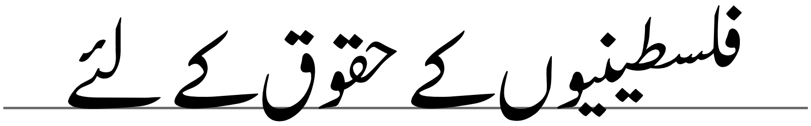


Shape change when joining to a preceding character

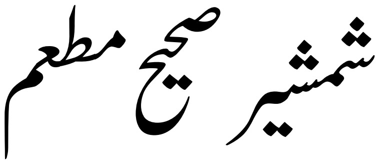


Shape change when occurring between characters

Figure 3.13: The issue of contextual shape change for a character. Dotted circles show the change in the shape of character p (*pay*) in the third figure.



A possible baseline

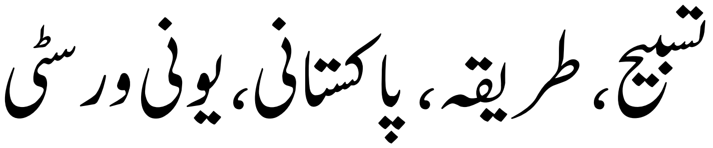


No baseline possible

Figure 3.14: Issue of random baseline in Nastaleeq. The issue of baseline determination is highlighted in the second row, where shape of the ligatures make it impossible to determine the accurate baseline.

height of a word or ligature depends upon the number and type of characters. Therefore, we see that a baseline can be determined for some text-lines, but for others it is not possible to do so (see Figure 3.14).

Huge Number of Ligatures: In Nastaleeq and in Naskh, a single word may consist of one or more ligatures. Some examples of words consisting of one, two, three, or fours ligatures are shown in Figure 3.15. The total number of unique ligatures in Urdu Nastaleeq is around 26,000 [KSKD15]. This amount makes it very hard to develop word-based or ligature-based holistic approaches for OCR. Though around 2000 ligatures cover more than 90% of the words in Urdu, classifying these 2000 ligatures is also a very difficult task from the perspective of Computer Vision (CV). The challenge becomes more trickier where two or more ligatures are differentiated by only a single dot or a diacritic mark.



تسبیح، طریقہ، پاکستانی، یونی ورسٹی

Figure 3.15: Some examples of Urdu ligatures. A word in Urdu may consist of one or more ligatures. The last word on the left consists of three ligatures and two individual letters as both of them are non-joinable either from preceding or succeeding letters.

Dots and Diacritics: Nabataean scripts are characterized by a lot of dots and diacritics marks above, below or inside a character. There could be up-to three dots associated with some characters in Urdu Nastaleeq script. Dot association is particularly very challenging w.r.t. OCR as their placement changes depending the shape of a character in a certain position. For some characters like چ (che) or ج (jeem), where a dot is present inside the character, the dot no longer remains inside as the shape of these character changes in the middle or beginning of a ligature (see Figure 3.16).

For neighboring characters appearing in a ligature, the dots may appear above or below of a different character. Figure 3.17 highlights this issue in a single text-line.

Scribed versus Printed Documents: The use of computers for creating Urdu content (books, newspapers, articles, magazines) is rather recent. First typesetting for Urdu Nastaleeq appeared in 1981 and prior to that most of the work was scribed by trained writers (locally known as *Katib*). A lot of Urdu historical literature is thus scribed. Some of the differences between scribed documents and the modern printed documents are described below:

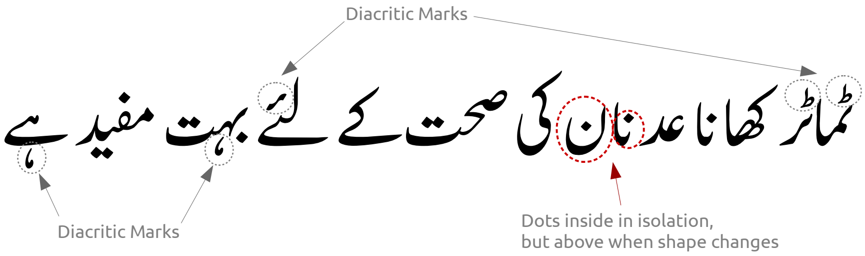

Diacritic Marks
ٹماٹر کھانا عدنان کی صحت کے لئے بہت مفید ہے
Diacritic Marks
Dots inside in isolation,
but above when shape changes

Figure 3.16: Importance of dots and diacritics. Dots and diacritics play an important role in the recognition of a letter. Some words are distinguished only by dots or diacritics.

- The shape of characters/ligatures does not remain consistent throughout the document. Figure 3.18 shows a scribed page. The circles in red show the same ligature. The difference in shapes of this ligature is evident.
- To preserve the text-line justification, the scribe may increase/stretch the ending ligature, thus making it very different to its usual form.
- The scribe may vertically stack part of a word for stylistic reasons and can also add decorations. This is a unique problem and has not been addressed so far.

3.6.2 Databases for Urdu Nastaleeq OCR

Urdu Nastaleeq OCR research is still in its infancy stage and there is still no commercial or open-source OCR system available for public use. However, there is an increasing trend among OCR researchers, especially those from the Indian sub-continent to develop a reliable Urdu OCR system. The first public Urdu dataset was proposed by Shafait et al. [SUHKB06]. It was a small dataset consisting of only 25 documents with

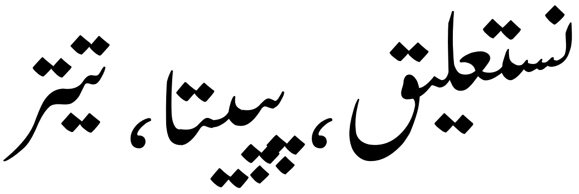
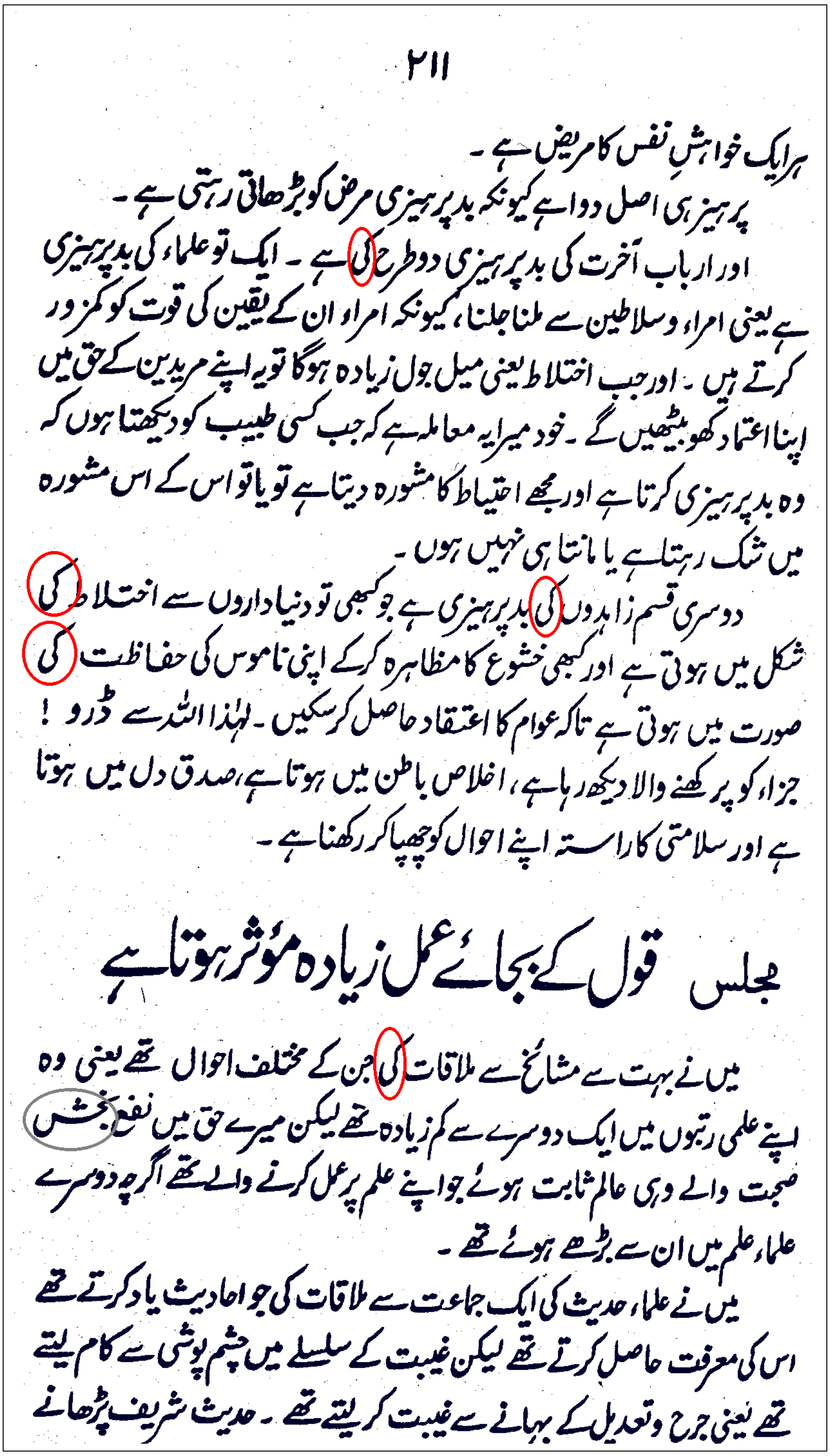


Figure 3.17: Effect of close proximity on the location of dots. The location of dots is disturbed due to the presence of characters in close proximity. These dots may appear above or below neighboring characters in a ligature (first word from right).



۲۱۱

ہر ایک خواہشِ نفس کا مریض ہے ۔
پرہیز ہی اصل دوا ہے کیونکہ بدپرہیزی مرض کو بڑھاتی رہتی ہے ۔
اور ارباب آخرت کی بدپرہیزی دو طرح کی ہے ۔ ایک تو علماء کی بدپرہیزی
ہے یعنی امراء و سلاطین سے ملنا جلنا، کیونکہ امراء ان کے یقین کی قوت کو کمزور
کرتے ہیں ۔ اور جب اختلاط یعنی میل جول زیادہ ہوگا تو یہ اپنے مریدین کے حق میں
اپنا اعتماد کھو بیٹھیں گے ۔ خود میرا یہ معاملہ ہے کہ جب کسی طبیب کو دیکھتا ہوں کہ
وہ بدپرہیزی کرتا ہے اور مجھے احتیاط کا مشورہ دیتا ہے تو یا تو اس کے اس مشورہ
میں شک رہتا ہے یا مانتا ہی نہیں ہوں ۔
دوسری قسم زاہدوں کی بدپرہیزی ہے جو کبھی تو دنیاداروں سے اختلاط کی
شکل میں ہوتی ہے اور کبھی خشوع کا مظاہرہ کرکے اپنی ناموس کی حفاظت کی
صورت میں ہوتی ہے تاکہ عوام کا اعتقاد حاصل کرسکیں ۔ لہٰذا اللہ سے ڈرو !
جزاء کو پرکھنے والا دیکھ رہا ہے، اخلاص باطن میں ہوتا ہے، صدق دل میں ہوتا
ہے اور سلامتی کا راستہ اپنے احوال کو چھپا کر رکھنا ہے ۔

مجلس قول کے بجائے عمل زیادہ مؤثر ہوتا ہے

میں نے بہت سے مشائخ سے ملاقات کی جن کے مختلف احوال تھے یعنی وہ
اپنے علمی رتبوں میں ایک دوسرے سے کم زیادہ تھے لیکن میرے حق میں نفع بخش
صحبت والے وہی عالم ثابت ہوئے جو اپنے علم پر عمل کرنے والے تھے اگرچہ دوسرے
علماء، علم میں ان سے بڑھے ہوئے تھے ۔
میں نے علماء، حدیث کی ایک جماعت سے ملاقات کی جو احادیث یاد کرتے تھے
اس کی معرفت حاصل کرتے تھے لیکن غیبت کے سلسلے میں چشم پوشی سے کام لیتے
تھے یعنی جرح و تعدیل کے بہانے سے غیبت کر لیتے تھے ۔ حدیث شریف پڑھانے

Figure 3.18: Urdu document scribed by a *Katib*. Red circles show the inconsistencies in shapes of the same ligature. The gray circle shows the stretching made to keep the justification of the text-line.

GT for both layout analysis and OCR. It contained pages from scribed documents, thereby making it hard to OCR.

Ijaz and Hussani [IH07] developed a large word dictionary containing around 50k unique words. They extracted words from the documents of six domains, namely, sports, news, finance, culture, entertainment, consumer information, and personal communication. The total number of words are over 18 million.

Urooj et al. published a large public database of Urdu Nastaleeq in different font-sizes [USHA14]. This database contains more than 100k words of Urdu. They developed this database from a lot of domains like letters, interviews, press, religion, sports, culture, health, science, short stories, translations, novels, and book reviews.

Sabbour et al. [SS13] published the *Urdu Printed Text Images (UPTI)* database. This dataset consists of over 10 thousand unique text-lines chosen from various resources. Each text-line is rendered synthetically with various degradation parameters. The database contains GT information at both text-line and ligature levels.

So far, we have discussed the issues and challenges that are faced in dealing with single-script documents. In the upcoming section, challenges that arise in developing OCR systems for documents with multiple scripts and languages will be described. These documents present unique challenges for recognition and therefore need separate treatment in processing.

3.7 Printed Documents with Multiple Scripts and Languages

In the present age of globalization, multilingual documents are highly prevalent. An increasing number of documents ranging from user manuals of electronic devices to travelling brochures, and from dictionaries to translations, use multiple languages to cater to a global audience. On the other hand, English has become an international language and the effects of this internationalization is evident in many contemporary articles. The challenges faced in developing OCR technology for multilingual documents can be best understood by classifying these document into various categories.

3.7.1 Categorization of Multilingual Documents

From the point of view of OCR, multilingual documents can be categorized into three categories and it is important to know the differences among the various kinds to fully understand the difficulties faced in their digitization.

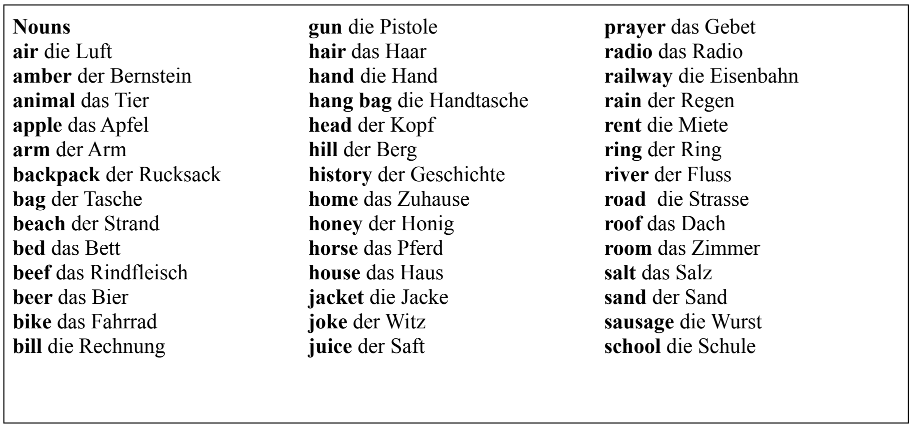
Same Script but Different Languages: All languages share the same family of script in these type of multilingual documents. Languages such as English, German, French, Italian share the same *Latin* script for writing just like Hindi and Nepali share the same script of *Devanagari* for writing. Figure 3.19-(a) shows an example of such documents.

Same Language but Different Scripts: Some languages can be written in multiple scripts. Urdu is one such language. It can be written in either *Nastaleeq* script or *Naskh* script. Another example is Japanese language, which can be written in *Hiragana*, *Kanji* or *Katakana* scripts. Figure 3.19-(b) shows an example of an Urdu document written in Naskh and Nastaleeq. Strictly speaking, these are not multilingual documents, but rather a document with multiple scripts; however, we would consider it as a type of 'multilingual' document in this thesis and use both the terms interchangeably.

Different Language and Different Scripts: The third category of multilingual documents is where both language and the script are different. Examples of this category are documents in English-Chinese, Arabic-French or Greek-Japanese. Figure 3.19-(c) shows an English-Arabic multilingual document.

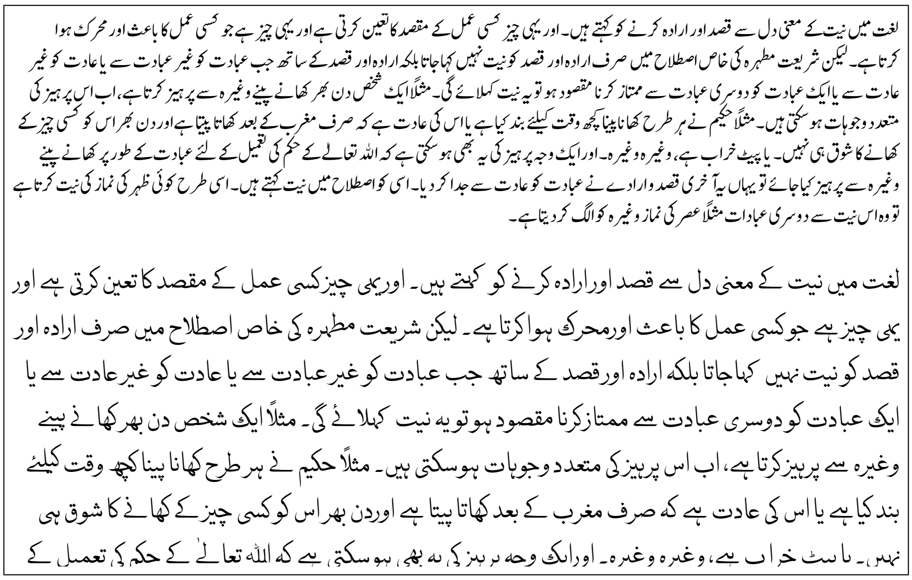
3.7.2 OCR Datasets for Multilingual Documents

MOCR has been traditionally carried out by first employing a script-identification step, so that the language-specific models can be used on recognized scripts. Available resources for script-identification are few. Most of the researchers have reported results on their private databases and they are not generally available for others [GDS10]. Kanungo et al. [KRM+05] proposed to use the Bible as the standard database to benchmark MOCR algorithms. However, this idea has not become popular.



Nouns
air die Luft
amber der Bernstein
animal das Tier
apple das Apfel
arm der Arm
backpack der Rucksack
bag der Tasche
beach der Strand
bed das Bett
beef das Rindfleisch
beer das Bier
bike das Fahrrad
bill die Rechnung
gun die Pistole
hair das Haar
hand die Hand
hang bag die Handtasche
head der Kopf
hill der Berg
history der Geschichte
home das Zuhause
honey der Honig
horse das Pferd
house das Haus
jacket die Jacke
joke der Witz
juice der Saft
prayer das Gebet
radio das Radio
railway die Eisenbahn
rain der Regen
rent die Miete
ring der Ring
river der Fluss
road die Strasse
roof das Dach
room das Zimmer
salt das Salz
sand der Sand
sausage die Wurst
school die Schule

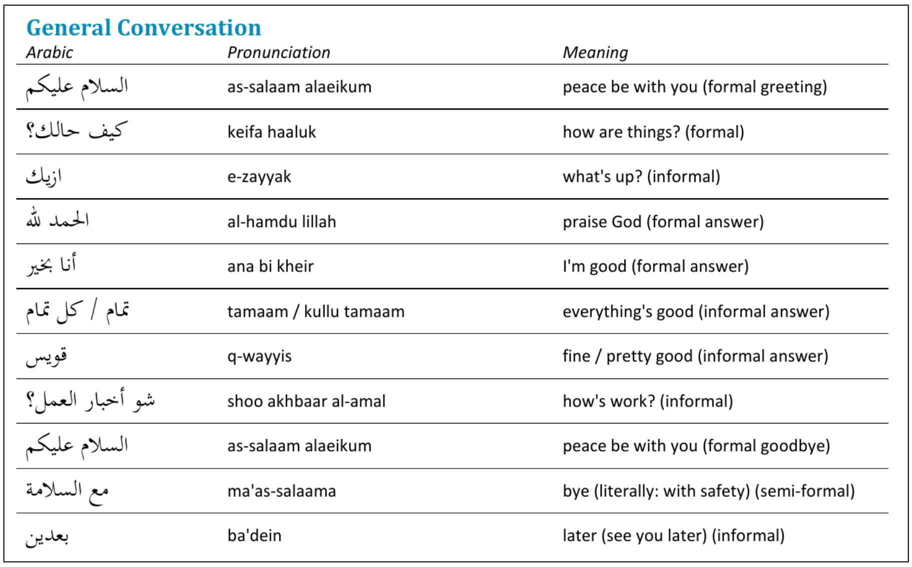
(a) Different languages same script



لغت میں نیت کے معنی دل سے قصد اور ارادہ کرنے کو کہتے ہیں۔ اور یہی چیز کسی عمل کے مقصد کا تعین کرتی ہے اور یہی چیز ہے جو کسی عمل کا باعث اور محرک ہوا
کرتا ہے۔ لیکن شریعت مطہرہ کی خاص اصطلاح میں صرف ارادہ اور قصد کو نیت نہیں کہا جاتا بلکہ ارادہ اور قصد کے ساتھ جب عبادت کو غیر عبادت سے یا عادت کو غیر
عادت سے یا ایک عبادت کو دوسری عبادت سے ممتاز کرنا مقصود ہو تو یہ نیت کہلائے گی۔ مثلاً ایک شخص دن بھر کھانے پینے وغیرہ سے پرہیز کرتا ہے، اب اس پرہیز کی
متعدد وجوہات ہو سکتی ہیں۔ مثلاً حکیم نے ہر طرح کھانا پینا کچھ وقت کیلئے بند کیا ہے یا اس کی عادت ہے کہ صرف مغرب کے بعد کھاتا پیتا ہے اور دن بھر اس کو کسی چیز کے
کھانے کا شوق ہی نہیں۔ یا پیٹ خراب ہے، وغیرہ وغیرہ۔ اور ایک وجہ پرہیز کی یہ بھی ہو سکتی ہے کہ اللہ تعالے کے حکم کی تعمیل کے لئے عبادت کے طور پر کھانے پینے
وغیرہ سے پرہیز کیا جائے تو یہاں یہ آخری قصد و ارادے نے عبادت کو عادت سے جدا کر دیا۔ اسی کو اصطلاح میں نیت کہتے ہیں۔ اسی طرح کوئی ظہر کی نماز کی نیت کرتا ہے
تو وہ اس نیت سے دوسری عبادات مثلاً عصر کی نماز وغیرہ کو الگ کر دیتا ہے۔

لغت میں نیت کے معنی دل سے قصد اورارادہ کرنے کو کہتے ہیں۔ اوریہی چیزکسی عمل کے مقصد کا تعین کرتی ہے اور
یہی چیز ہے جوکسی عمل کا باعث اورمحرك ہوا کرتا ہے۔ لیکن شریعت مطہرہ کی خاص اصطلاح میں صرف ارادہ اور
قصد کو نیت نہیں کہا جاتا بلکہ ارادہ اورقصد کے ساتھ جب عبادت کو غیرعبادت سے یا عادت کو غیرعادت سے یا
ایك عبادت کو دوسری عبادت سے ممتازکرنا مقصود ہوتو یہ نیت کہلائے گی۔ مثلاً ایك شخص دن بھر کھانے پینے
وغیرہ سے پرہیزکرتا ہے، اب اس پرہیزکی متعدد وجوہات ہوسکتی ہیں۔ مثلاً حکیم نے ہرطرح کھانا پینا کچھ وقت کیلئے
بند کیا ہے یا اس کی عادت ہے کہ صرف مغرب کے بعد کھاتا پیتا ہے اوردن بھراس کو کسی چیز کے کھانے کا شوق ہی
نہیں۔ یا پیٹ خراب ہے، وغیرہ وغیرہ۔ اورایك وجہ پرہیز کی یہ بھی ہو سکتی ہے کہ اللہ تعالےٰ کے حکم کی تعمیل کے

(b) One language different scripts



General Conversation

Arabic	*Pronunciation*	*Meaning*
السلام عليكم	as-salaam alaeikum	peace be with you (formal greeting)
كيف حالك؟	keifa haaluk	how are things? (formal)
ازيك	e-zayyak	what's up? (informal)
الحمد لله	al-hamdu lillah	praise God (formal answer)
أنا بخير	ana bi kheir	I'm good (formal answer)
تمام / كل تمام	tamaam / kullu tamaam	everything's good (informal answer)
قويس	q-wayyis	fine / pretty good (informal answer)
شو أخبار العمل؟	shoo akhbaar al-amal	how's work? (informal)
السلام عليكم	as-salaam alaeikum	peace be with you (formal goodbye)
مع السلامة	ma'as-salaama	bye (literally: with safety) (semi-formal)
بعدين	ba'dein	later (see you later) (informal)

(c) Different languages different scripts

Figure 3.19: Three main categories of multilingual documents.

3.8 Chapter Summary

This chapter gives an overview of difficulties and challenges posed by various languages and scripts for the purpose of OCR. The OCR for complex scripts is challenging and new algorithms are required to address these challenges. While most of the modern Latin scripts enjoy multitude of resources for OCR, the scarcity of such resources render the OCR very challenging for historical scripts, many modern ‘underprivileged’ scripts, and multilingual documents.

Chapter 4

Benchmark Datasets for OCR

Numerous character recognition algorithms require sizable ground-truthed real-world data for training and benchmarking. The quantity and quality of training data directly affects the generalization accuracy of a trainable OCR model. However, developing GT data manually is overwhelmingly laborious, as it involves a lot of effort to produce a reasonable database that covers all possible words of a language. Transcribing historical documents is even more gruelling as it requires language expertise in addition to manual labelling efforts. The increased human efforts give rise to financial aspects of developing such datasets and could restrict the development of large-scale annotated databases for the purpose of OCR. It has been pointed out in the previous chapter that scarcity of training data is one of the limiting factors in developing reliable OCR systems for many historical as well as for some modern scripts.

The challenge of limited training data has been overcome by the following contributions of this thesis:

- A semi-automated methodology to generate the GT database for cursive scripts at ligature level has been proposed. This methodology can equally be applied to produce character-level GT data. Section 4.2 reports the specifics of this method for cursive Nabataean scripts.
- Synthetically generated text-line databases have been developed to enhance the OCR research. These datasets include a database for Devanagari script (Deva-DB), a subset of printed Polytonic Greek script (Polytonic-DB), and three datasets for Multilingual OCR (MOCR) tasks. Section 4.3 details this process and describes the fine points about these datasets.

4.1 Related Work

There are basically two types of methodologies that have been proposed in the literature. The first is to extract identifiable symbols from the document image and apply some clustering methods to create representative prototypes. These prototypes are then assigned text labels. The second approach is to synthesize the document images from the textual data. These images are degraded using various image defect models to reflect the scanning artifacts. These degradation models [Bai92] include *resolution, blur, threshold, sensitivity, jitter, skew, size, baseline*, and *kerning*. Some of these artifacts are discussed in Section 4.3 where they are used to generate text-line images from the text.

The use of synthesized training data is increasing and there are many datasets reported in the literature using this methodology. One dataset that is prominent among these types is the *Arabic Printed Text Images (APTI)* database, which is proposed by Slimane et al. [SIK+09]. This database is synthetically generated covering ten different Arabic fonts and as many font-sizes (ranging from 6 to 24). It is generated from various Arabic sources and contains over 1 million words. The number increases to over 45 million words when rendered using ten fonts, four styles and ten font-sizes.

Another example of a synthetic text-line image database is the *Urdu Printed Text Images (UPTI)* database, published by Sabbour and Shafait [SS13]. This dataset consists of over 10 thousand unique text-lines selected from various sources. Each text-line is rendered synthetically with various degradation parameters. Thus the actual size of the database is quite large. The database contains GT information at both text-line and ligature levels.

The second approach in automating the process of generating an OCR database from scanned document images is to find the alignment of the transcription of the text lines with the document image. Kanungo et al. [KH99] presented a method for generating character GT automatically for scanned documents. A document is first created electronically using any typesetting system. It is then printed out and scanned. Next, the corresponding feature points from both versions of the same document are found and the parameters of the transformation are estimated. The ideal GT information is transformed accordingly using these estimates. An improvement in this method is proposed by Kim and Kanungo [KK02] by using an attributed branch-and-bound algorithm.

Von Beusekom et al. [vBSB08] proposed a robust and pixel-accurate alignment method. In the first step, the global transformation parameters are estimated in a similar manner as in [KK02]. In the second step, the adaptation of the smaller region is carried out.

Pechwitz et al. [PMM+02] presented the IfN/ENIT database of handwritten Arabic names of cities along with their postal codes. A projection profile method is used to extract words and the postal codes automatically. Mozaffari et al. [MAM+08] developed a similar database (IfN/Farsi-database) for handwritten Farsi (Persian) names of cities. Sagheer et al. [SHNS09] also proposed a similar methodology for generating an Urdu database for handwriting recognition.

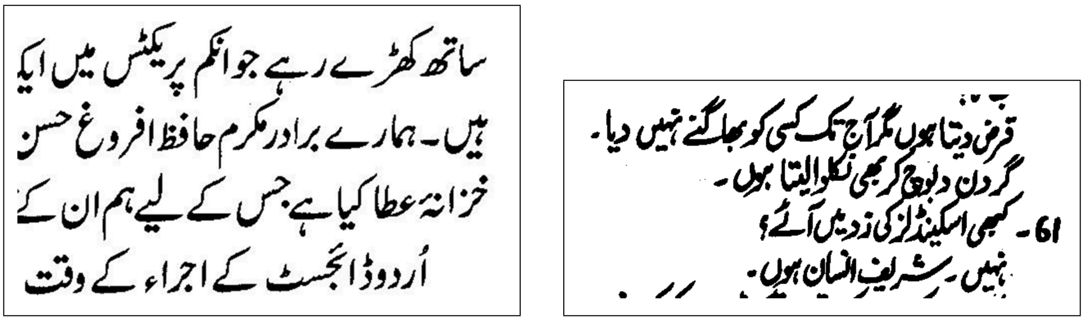
Vamvakas et al. [VGSP08] proposed that a character database for historical documents may be constructed by choosing a small subset of images and then using character segmentation and clustering techniques. This work is similar to our approach; however, the main difference is the use of a different segmentation technique for Urdu ligatures and the utilization of a dissimilar clustering algorithm.

4.2 Semi-automated Database Generation

This section describes an approach to automate the process of OCR database preparation from scanned documents. It is believed that the proposed automation will greatly reduce the manual efforts required to develop OCR databases for cursive scripts. The basic idea is to apply ligature-clustering prior to manual labeling. Two prototype datasets for Urdu Nastaleeq script have been developed using the proposed technique.

Urdu belongs to the family of cursive scripts where words mainly consist of ligatures. Ligatures are formed by joining individual characters and the shape of a character in a ligature depends on its position within the word. Moreover, there are dots and diacritics that are associated with certain characters. Each ligature in Urdu is separated from other ligatures or its own diacritics by vertical, horizontal or diagonal (slanted) space. The properties of this script along with the challenges it poses for OCR have been described in Section 3.6. It is assumed in the context of the present section that the smallest unit of the script is a ligature, which may either be a combination of many characters or a single non joinable character. There are around 26,000 ligatures in Urdu Nastaleeq script, and a reasonable database must cover all of them.

The method to semi automatically generate a database for Urdu Nastaleeq starts with binarization as the pre-processing step. Urdu ligatures are then extracted from the text images. These ligatures are then clustered prior to manual labeling of the



ساتھ کھڑے رہے جوانکم پریکٹس میں ایک
ہیں۔ ہمارے برادر مکرم حافظ افروغ حسن
خزانہ عطا کیا ہے جس کے لیے ہم ان کے
اُردو ڈائجسٹ کے اجراء کے وقت

(a) An example of clean Urdu Documents

قرض دیتا ہوں مگر آج تک کسی کو بھاگنے نہیں دیا۔
گردن دبوچ کر بھی نکلوا لیتا ہوں۔
61۔ کبھی اسکینڈلز کی زد میں آئے؟
نہیں۔ شریف انسان ہوں۔

(b) A document with narrow line spacing

Figure 4.1: Examples of ligature-based Urdu Nastaleeq script.

correct ligatures. The following sub-sections present a detailed view of the proposed method.

4.2.1 Preprocessing

Binarization is the only preprocessing step in the proposed method; however, skew detection and correction may be included as further preprocessing steps. Local thresholding technique [SP00] is used for the binarization purpose. Fast implementation of this algorithm proposed by Shafait et al. [SKB08] has been used to speed-up the process. Two parameters, namely local window size and k-parameter, are needed to be set empirically according to the documents. The local window size is taken to be 70 × 70 and k-parameter is set to 0.3.

4.2.2 Ligature Extraction

Ligature extraction may be carried out in two ways: one is to apply ligature extraction algorithm directly on the binarized image, while the second is to extract text-lines before applying ligature extraction. The former is suitable where documents are clean having well-defined text-line spacing (see Figure 4.1-(a)) and the latter is suitable when text-lines are not separated very well in the documents (Figure 4.1-(b)), and in case of degraded historical documents.

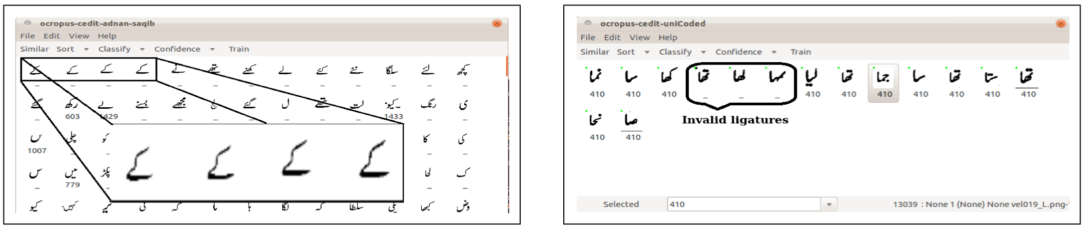
Narrow line separation results in poor connected component analysis; thereby, many ligatures from the text-lines above and below merge together. The decision to apply text-line segmentation is taken on the basis of line-spacing in a particular document. Ligature extraction is started by applying connected component analysis. The list of connected components is first divided into two parts, base components and diacritics (including dots). This division is based on connected component's height, their width, and the ratio of the two. In the context of this chapter, font variations are not considered and the primary focus is to cover the typically used fonts in Urdu books and magazines. Therefore, thresholds for separating main ligatures and diacritics are set empirically on primary font's size and they remain same for all the document images in our dataset.

It is not possible to separate Urdu ligatures by a single threshold value. Therefore, different thresholds have been employed as per properties of a particular ligature. For ligature consisting of single 'آ' (*alif*), the average height to width ratio is 4.0 and the average width of this ligature is around 6 pixels. For ligatures like 'ب' (*bay*), 'ت' (*tay*) and 'ث' (*say*), the average height to width ratio is 0.4 and the average width is around 30 pixels. For all other ligatures, it is sufficient to employ a width greater than 10 pixels.

If there are no diacritics in a ligature, e.g., in 'لو', then no further processing is needed. However, if one or more diacritics are present, e.g., in 'نستعلیق', then these diacritics must be associated to the base component to completely extract a ligature. Diacritics are searched in the neighborhood of a base component by extending the bounding box of the base connected component. This window size depends on the font size; but since we have used documents only with the dominant font size, this window is set according to that font size. Presently, the bounding box of the base component is extended by 15 pixels on top and bottom and by 10 pixels on right and left. Ligatures extracted in this manner are then saved to a database file for further clustering and labeling.

4.2.3 Clustering

As it has been mentioned that due to the huge amount of ligatures that are present in a cursive script, the labeling of individual ligatures becomes highly impractical. Hence, it is proposed that the extracted ligatures be clustered according to similar shapes. For the purpose of clustering, the epsilon-net clustering technique is employed. By simply changing the value of epsilon, we can control the number of clusters. The value for epsilon is set empirically to get moderate amount of clusters, so that they can


ocropus-cedit-adnan-saqib
File Edit View Help
Similar Sort Classify Confidence Train
603
429
1433
1007
779
ocropus-cedit-uniCoded
File Edit View Help
Similar Sort Classify Confidence Train
410
Invalid ligatures
Selected
410
13039 : None 1 (None) None vel019_L.png-

(a) Cluster representatives (b) Validating an individual cluster

Figure 4.2: The graphical interface to examine the clusters and to assign the unique identification.

be managed easily at the manual step of validation. The features used for epsilon clustering are bit maps of the ligatures. Moreover, this method is relatively faster than the k-means clustering.

4.2.4 Ligature Labeling

The next step is to verify the clusters and modify them manually, if needed. The OCRopus framework provides a proficient graphical user interface to do this without much hassle. It is also possible that the clustering divides a single ligature in more than one cluster (see Figure 4.2-(a)), hence one needs to merge different clusters to save time at a latter stage of labeling. Moreover, one can also modify the step of dividing a cluster in a way to retain only valid members (same label as that of representative), assign null class to incorrect members and then apply further iterations of clustering on null class. In the current work, merging of same ligature clusters precedes the manual labeling and only single cluster iteration is employed. After this verification step, each cluster is examined individually to identify invalid clusters, which are then discarded. Again OCRopus framework is used for this purpose (see Figure 4.2-(b)).

At the end of this labeling process, we have a database whose entries indicate the following information about a ligature:

- Image file name from where this ligature was originally extracted.
- Bounding box information regarding the location of a ligature in the document image.
- Unicode string corresponding to the character forming this ligature.

4.2.5 Experiments and Results

This section describes the experimental setup and the evaluation of the results. Two prototype datasets for Urdu script have been developed using the proposed technique. One dataset consists of clean documents such as that shown in Figure 2-(a). At present, only 20 such document images have been used. Here, this dataset is referred to as DB-I. The second dataset (referred to as DB-II) consists of 15 documents written by a calligrapher such as that shown in Figure 4.1-(b). An important property of calligraphic documents is that the shape of a ligature does not remain identical within the document and minor differences in their shapes may remain throughout the document. GT information about the DB-II is available which is used to gauge the accuracy of line segmentation algorithm. The importance of choosing these two datasets is to evaluate upper and lower bounds on the performance of the proposed algorithm. Performance evaluation metric used in the present work is **ligature coverage**, which refers to the number of ligatures in the dataset that are correctly labeled by the clustering step, followed by the manual validation step.

The ligature extraction algorithm is able to find 16,857 ligatures in DB-I database. The epsilon-net based clustering, then clusters these ligatures into 778 clusters. Each individual cluster is subsequently examined to verify the clustering and the Unicode values are assigned to new clusters. The invalid ligatures are discarded at this point. The ligature coverage achieved by this process is 82.3%. This high ligature coverage is due to sufficient line spacing and non-touching ligatures.

The inherent difficulty with any connected-component analysis based method is the poor accuracy in case of overlapping lines and touching ligatures. To solve this problem in DB-II dataset, a line segmentation algorithm [BSB11] is employed. The segmentation accuracy of this algorithm is over 90%. The second problem of touching ligatures may be improved by using more sophisticated techniques. However, we are not interested in fine separation of individual ligatures as the errors may be corrected at latter manual labeling stage. Hence, we did not tackle this problem in this work.

From DB-II database, a total of 18,914 ligatures are extracted. Then the clustering of these ligatures resulted in 1,132 clusters. After the labeling process, the total ligature coverage is around 62.7%. Inconsistency in ligatures' shape due to the handwriting of the calligrapher results in poor clustering accuracy for DB-II dataset. In this case, simple shape-based clustering methods do not work sufficiently and other methods are needed to be explored.

4.2.6 Conclusion

A semi-automated methodology is proposed to generate a GT database from scanned documents for cursive scripts at ligature level. The same methodology can be used for rapid generation of character level datasets for other scripts as well. One unsatisfactory aspect of this methodology is the use of heuristics to extract ligatures from the document images. These heuristics need to be adapted accordingly for other scripts. It is also observed that the performance of this method is directly affected by the choice of segmentation method and the quality of document images.

The second approach to develop a large-scale GT OCR database is to use the image degradation models. This approach is described in the following section.

4.3 Synthetic Text-Line Generation

The use of artificial data is getting popular in computer vision domain for object recognition purpose. A similar path is taken in this thesis to address the issue of limited GT data. Baird [Bai92] proposed several degradation models to generate artificial data from the text (ASCII) form. There are many parameters that can be altered to make the artificially generated text-line images resemble closely to those obtained from a scanning process. Some of the significant parameters are:

Blur: It is the pixel-wise spread in the output image, and is modeled as circular Gaussian filter.

Threshold: It is used to distort the image by randomly removing the text pixels. If a pixel value is greater than this threshold, then it is a background pixel.

Size: It is the height and width of individual characters in the image. It is modeled by image scaling operations.

Skew: It is the rotation angle of the output symbol. The resulting angle is skewed to right or left by specifying the ‘skew’ parameter.

In this thesis, a utility based on these degradation models from OCRopus [OCR15] (open-source OCR framework) is used to generate the artificial data. The aforesaid OCRopus utility requires *utf-8*-encoded text-lines to generate the corresponding text-line images along with the *ttf*-type fontfiles. The process of line image generation is shown in Figure 4.3. The user can specify the parameter values or use the default values.

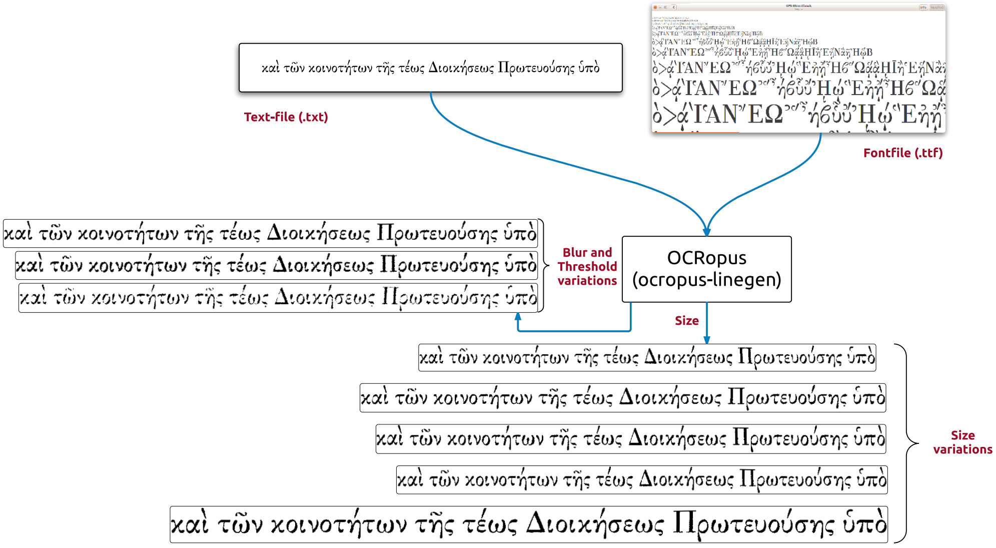

καὶ τῶν κοινοτήτων τῆς τέως Διοικήσεως Πρωτευούσης ὑπὸ
Text-file (.txt)
Fontfile (.ttf)
Blur and Threshold variations
OCRopus (ocropus-linegen)
Size
Size variations

Figure 4.3: The synthetic text-line image generation process using various degradation parameters. With a textual input along with the desired *fontfile*, various degradation parameters can be tuned to generate the desired output.

4.4 OCR Databases

This section lists various datasets that have been developed using the synthetic generation process as described in the previous section. These datasets are available freely for research purposes and can be obtained from the author.

4.4.1 Deva-DB – OCR Database for Devanagari Script

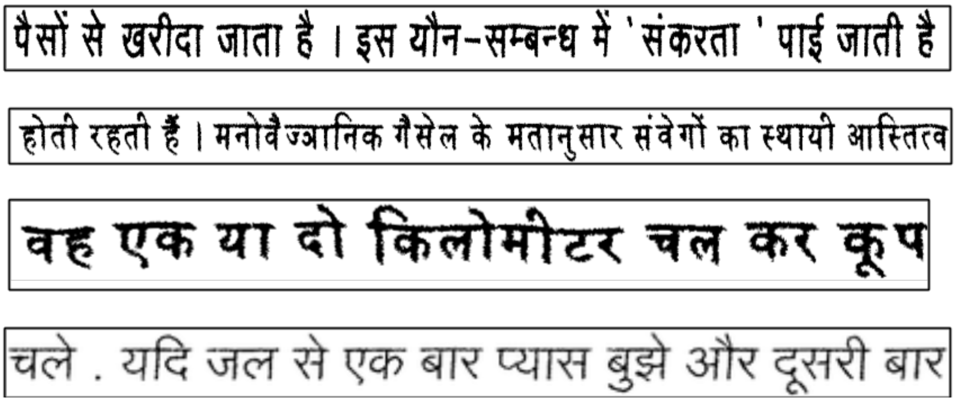
A new database of Devanagari script is presented[1] to advance the OCR research in Devanagari OCR. It can provide a common platform for researchers in this domain to benchmark their algorithms.

This database, named **Deva-DB** consists of two parts. The first part is the text-lines taken from the work of Setlur et al. [KNSG05]. The GT information is represented in transliteration form in this work. We have transcribed 621 text-lines manually in standard Unicode form. This data is used for evaluation purposes only, that is to test the LSTM model trained with the artificial data.

Second part of this database is the **synthetically** generated text-line images using OCRopus framework. These text-lines were chosen from various on-line resources covering the fields of current-affairs, religion, classical literature and science. Some sample images from this database are shown in Figure 4.4.

---

[1] Prepared by the first author of research work reported in [KUHB15]



पैसों से खरीदा जाता है । इस यौन-सम्बन्ध में 'संकरता ' पाई जाती है

होती रहती हैं । मनोवैज्ञानिक गैसेल के मतानुसार संवेगों का स्थायी आस्तित्व

वह एक या दो किलोमीटर चल कर कूप

चले . यदि जल से एक बार प्यास बुझे और दूसरी बार

Figure 4.4: Four sample images taken from the training set. These are synthetic line images generated using different fonts and different degradation models.

To check the quality of the training set, a comparison is made between the character and word statistics of this set with that published by Chaudhuri et al. [CP97]. The ten most frequently used characters in Hindi based on three million Hindi words are shown in Table 4.1. We collected a similar statistic for our training data based on approximately one million words. As seen from Table 4.1, the relative frequency of the characters in the proposed training set is similar to that provided by Chaudhuri et al. [CP97] [CP97]. IIIT Hyderabad, India [III14] has published a word frequency from a Hindi language corpus containing three million words. The proposed training set also matches the top ten frequent words of that work.

Table 4.1: Comparison of the character frequency in our training data set. The left side shows the frequency of characters as published by [CP97] and the right side shows the character frequency in our set. The frequency of most common characters in our set is quite close to the frequency of characters published by [CP97]

Symbol	Occurence(%)	Symbol	Occurence(%)
◌ा	10.12	◌ा	8.33
क	7.94	क	6.81
र	7.40	र	6.07
◌े	6.53	◌े	5.13
न	5.06	त	4.22
◌ी	4.47	ि◌	4.19
ह	4.39	न	4.08
स	4.36	स	3.93
त	4.32	ह	3.50
ि◌	4.22	◌ी	3.33

4.4.2 Polyton-DB – Greek Polytonic Database

A collection of some transcribed data is available under OldDocPro project [GSL$^+$][2], for the recognition of machine-printed and handwritten polytonic documents. The big issue of large amount of transcribed data for training is overcome by the use of synthetic data (generated using OCRopus framework). The contribution of this thesis is the introduction of synthetic text-line images as part of the freely available Greek Polytonic Database, called *Polyton-DB*[3], which includes printed polytonic Greek documents from three main sources:

Greek Parliament Proceedings: This database (a sample shown in Figure 4.5-a) contains 3203 text-lines images, that are taken from 33 scanned pages of the Greek Parliament proceedings. The documents used in this database correspond to the speeches of Greek politicians of various eras in the 19^{th} and the 20^{th} centuries.

Greek official government Gazette: These documents consists of 687 text-line images (a sample shown in Figure 4.5-b), which are picked from five scanned pages of Greek Official Government Gazette.

Appian's Roman History: 315 documents from Appians's Roman history (written in Greek language before AD 165) are used to create this database (a sample shown in Figure 4.5-c). It contains 11,799 text-line images. Appian's Roman history scans are clean images which is not the usual case with historical documents. Moreover, the writing style is different than the other two resources. To better train the OCR models for historical documents, synthetically degraded text-lines are generated from this corpus.

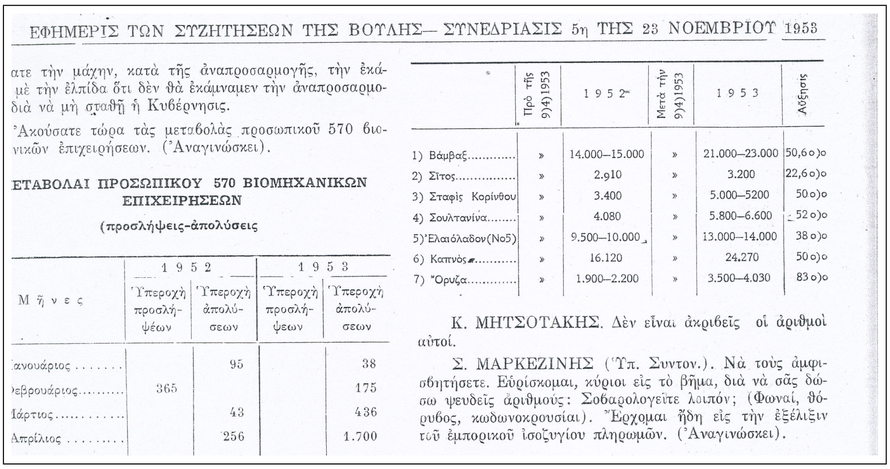
4.4.3 Databases for Multilingual OCR (MOCR)

MOCR is a relatively new field of research and there are not many publicly available databases to test the OCR algorithms in this regard. Moreover, the efforts till now have focused on script identification instead of complete processing of such documents. Most of the reported works have utilized datasets that are either private or no longer available. There are three main contributions of this thesis to test the MOCR algorithms.

---

[2] `http://www.iit.demokritos.gr/~nstam/GRPOLY-DB`

[3] collection available from `http://media.ilsp.gr/PolytonDB`



ΕΦΗΜΕΡΙΣ ΤΩΝ ΣΥΖΗΤΗΣΕΩΝ ΤΗΣ ΒΟΥΛΗΣ— ΣΥΝΕΔΡΙΑΣΙΣ 5η ΤΗΣ 23 ΝΟΕΜΒΡΙΟΥ 1953

ατε τὴν μάχην, κατὰ τῆς ἀναπροσαρμογῆς, τὴν ἐκά-
μὲ τὴν ἐλπίδα ὅτι δὲν θὰ ἐκάμναμεν τὴν ἀναπροσαρμο-
διὰ νὰ μὴ σταθῇ ἡ Κυβέρνησις.

Ἀκούσατε τώρα τὰς μεταβολὰς προσωπικοῦ 570 βιο-
νικῶν ἐπιχειρήσεων. (Ἀναγινώσκει).

ΕΤΑΒΟΛΑΙ ΠΡΟΣΩΠΙΚΟΥ 570 ΒΙΟΜΗΧΑΝΙΚΩΝ
ΕΠΙΧΕΙΡΗΣΕΩΝ

(προσλήψεις-ἀπολύσεις

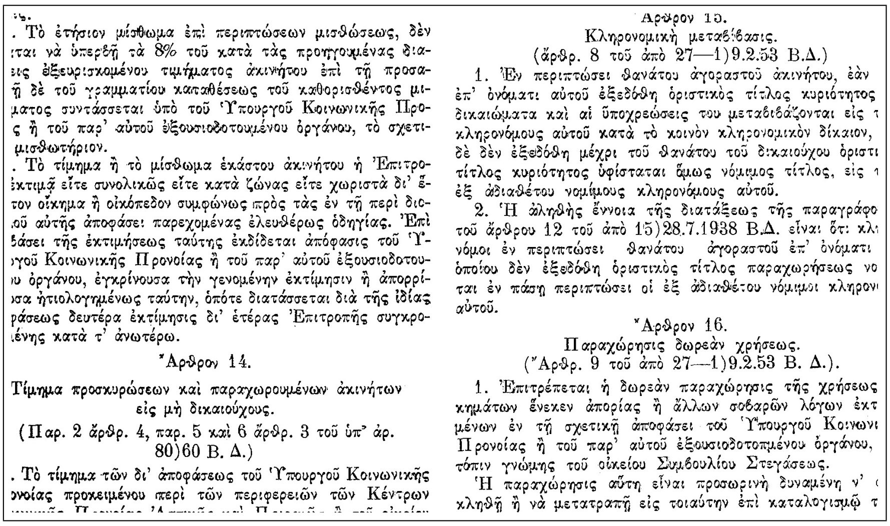
Μῆνες	1952		1953	
	Ὑπεροχὴ προσλή-ψέων	Ὑπεροχὴ ἀπολύ-σεων	Ὑπεροχὴ προσλή-ψεων	Ὑπεροχὴ ἀπολύ-σεων
ανουάριος		95		38
εβρουάριος	365			175
άρτιος		43		436
Απρίλιος		256		1.700

	Πρὸ τῆς 9)4)1953	1952	Μετὰ τὴν 9)4)1953	1953	Αὔξησις
1) Βάμβαξ............	»	14.000–15.000	»	21.000–23.000	50,6 ο)ο
2) Σῖτος..............	»	2.910	»	3.200	22,6 ο)ο
3) Σταφὶς Κορίνθου	»	3.400	»	5.000–5200	50 ο)ο
4) Σουλτανίνα........	»	4.080	»	5.800–6.600	52 ο)ο
5) Ἐλαιόλαδον(Νο5)	»	9.500–10.000	»	13.000–14.000	38 ο)ο
6) Καπνὸς..........	»	16.120	»	24.270	50 ο)ο
7) Ὄρυζα............	»	1.900–2.200	»	3.500–4.030	83 ο)ο

Κ. ΜΗΤΣΟΤΑΚΗΣ. Δὲν εἶναι ἀκριβεῖς οἱ ἀριθμοὶ αὐτοί.

Σ. ΜΑΡΚΕΖΙΝΗΣ (Ὑπ. Συντον.). Νὰ τοὺς ἀμφισβητήσετε. Εὑρίσκομαι, κύριοι εἰς τὸ βῆμα, διὰ νὰ σᾶς δώσω ψευδεῖς ἀριθμούς: Σοβαρολογεῖτε λοιπόν; (Φωναί, θόρυβος, κωδωνοκρουσίαι). Ἔρχομαι ἤδη εἰς τὴν ἐξέλιξιν τοῦ ἐμπορικοῦ ἰσοζυγίου πληρωμῶν. (Ἀναγινώσκει).

(a) Parliament proceedings



. Τὸ ἐτήσιον μίσθωμα ἐπὶ περιπτώσεων μισθώσεως, δὲν
ιται νὰ ὑπερβῇ τὰ 8% τοῦ κατὰ τὰς προηγουμένας δια-
εις ἐξευρισκομένου τιμήματος ἀκινήτου ἐπὶ τῇ προσα-
ῇ δὲ τοῦ γραμματίου καταθέσεως τοῦ καθορισθέντος μι-
ματος συντάσσεται ὑπὸ τοῦ Ὑπουργοῦ Κοινωνικῆς Προ-
ς ἢ τοῦ παρ' αὐτοῦ ἐξουσιοδοτουμένου ὀργάνου, τὸ σχετι-
μισθωτήριον.

. Τὸ τίμημα ἢ τὸ μίσθωμα ἑκάστου ἀκινήτου ἡ Ἐπιτρο-
ἐκτιμᾷ εἴτε συνολικῶς εἴτε κατὰ ζώνας εἴτε χωριστὰ δι' ἕ-
τον οἴκημα ἢ οἰκόπεδον συμφώνως πρὸς τὰς ἐν τῇ περὶ διο-
οῦ αὐτῆς ἀποφάσει παρεχομένας ἐλευθέρως ὁδηγίας. Ἐπὶ
βάσει τῆς ἐκτιμήσεως ταύτης ἐκδίδεται ἀπόφασις τοῦ Ὑ-
γοῦ Κοινωνικῆς Προνοίας ἢ τοῦ παρ' αὐτοῦ ἐξουσιοδοτου-
υ ὀργάνου, ἐγκρίνουσα τὴν γενομένην ἐκτίμησιν ἢ ἀπορρί-
σα ἠτιολογημένως ταύτην, ὁπότε διατάσσεται διὰ τῆς ἰδίας
ράσεως δευτέρα ἐκτίμησις δι' ἑτέρας Ἐπιτροπῆς συγκρο-
ένης κατὰ τ' ἀνωτέρω.

Ἄρθρον 14.

Τίμημα προσκυρώσεων καὶ παραχωρουμένων ἀκινήτων
εἰς μὴ δικαιούχους.

(Παρ. 2 ἄρθρ. 4, παρ. 5 καὶ 6 ἄρθρ. 3 τοῦ ὑπ' ἀρ.
80)60 Β. Δ.)

. Τὸ τίμημα τῶν δι' ἀποφάσεως τοῦ Ὑπουργοῦ Κοινωνικῆς
νοίας προκειμένου περὶ τῶν περιφερειῶν τῶν Κέντρων

Ἄρθρον 15.

Κληρονομικὴ μεταβίβασις.

(ἄρθρ. 8 τοῦ ἀπὸ 27—1)9.2.53 Β.Δ.)

1. Ἐν περιπτώσει θανάτου ἀγοραστοῦ ἀκινήτου, ἐὰν
ἐπ' ὀνόματι αὐτοῦ ἐξεδόθη ὁριστικὸς τίτλος κυριότητος
δικαιώματα καὶ αἱ ὑποχρεώσεις του μεταβιβάζονται εἰς τ
κληρονόμους αὐτοῦ κατὰ τὸ κοινὸν κληρονομικὸν δίκαιον,
δὲ δὲν ἐξεδόθη μέχρι τοῦ θανάτου τοῦ δικαιούχου ὁριστι
τίτλος κυριότητος ὑφίσταται ὅμως νόμιμος τίτλος, εἰς τ
ἐξ ἀδιαθέτου νομίμους κληρονόμους αὐτοῦ.

2. Ἡ ἀληθὴς ἔννοια τῆς διατάξεως τῆς παραγράφο
τοῦ ἄρθρου 12 τοῦ ἀπὸ 15)28.7.1938 Β.Δ. εἶναι ὅτι κλ
νόμοι ἐν περιπτώσει θανάτου ἀγοραστοῦ ἐπ' ὀνόματι
ὁποίου δὲν ἐξεδόθη ὁριστικὸς τίτλος παραχωρήσεως νο
ται ἐν πάσῃ περιπτώσει οἱ ἐξ ἀδιαθέτου νόμιμοι κληρον
αὐτοῦ.

Ἄρθρον 16.

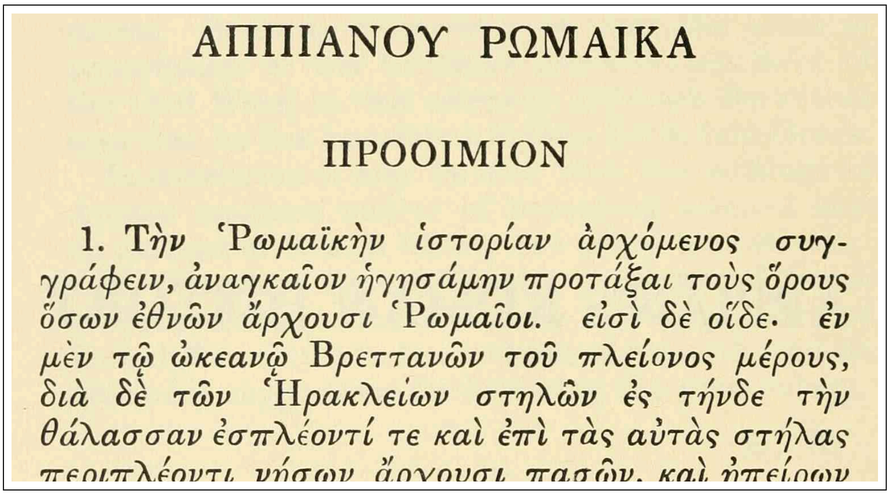
Παραχώρησις δωρεὰν χρήσεως.

(Ἄρθρ. 9 τοῦ ἀπὸ 27—1)9.2.53 Β. Δ.).

1. Ἐπιτρέπεται ἡ δωρεὰν παραχώρησις τῆς χρήσεως
κτημάτων ἕνεκεν ἀπορίας ἢ ἄλλων σοβαρῶν λόγων ἐκτ
μένων ἐν τῇ σχετικῇ ἀποφάσει τοῦ Ὑπουργοῦ Κοινωνι
Προνοίας ἢ τοῦ παρ' αὐτοῦ ἐξουσιοδοτουμένου ὀργάνου,
τόπιν γνώμης τοῦ οἰκείου Συμβουλίου Στεγάσεως.

Ἡ παραχώρησις αὕτη εἶναι προσωρινὴ δυναμένη ν'
κληθῇ ἢ νὰ μετατραπῇ εἰς τοιαύτην ἐπὶ καταλογισμῷ τ

(b) Govt. gazette



ΑΠΠΙΑΝΟΥ ΡΩΜΑΙΚΑ

ΠΡΟΟΙΜΙΟΝ

1. Τὴν Ῥωμαϊκὴν ἱστορίαν ἀρχόμενος συγγράφειν, ἀναγκαῖον ἡγησάμην προτάξαι τοὺς ὅρους ὅσων ἐθνῶν ἄρχουσι Ῥωμαῖοι. εἰσὶ δὲ οἵδε. ἐν μὲν τῷ ὠκεανῷ Βρεττανῶν τοῦ πλείονος μέρους, διὰ δὲ τῶν Ἡρακλείων στηλῶν ἐς τήνδε τὴν θάλασσαν ἐσπλέοντί τε καὶ ἐπὶ τὰς αὐτὰς στήλας περιπλέοντι νήσων ἄρχουσι πασῶν, καὶ ἠπείρων

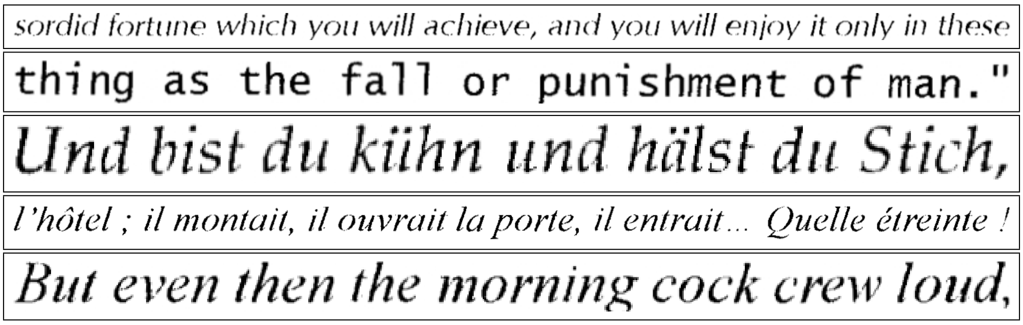
(c) Appian's history

Figure 4.5: Sample documents from three main sources in Polyton-DB.

- A database to gauge the cross-language performance of any OCR algorithm. The text-lines in this corpus are generated synthetically using the procedure

Table 4.2: Number of text-line images in each of English, French, German and mix-script datasets.

Language	Total Text-lines	Training	Test
English	85,350	81,600	4750
French	85,350	81,600	4750
German	114,749	110,400	4349
Mixed-Data	85,350	81,600	4750



sordid fortune which you will achieve, and you will enjoy it only in these

thing as the fall or punishment of man."

Und bist du kühn und hälst du Stich,

l'hôtel ; il montait, il ouvrait la porte, il entrait... Quelle étreinte !

But even then the morning cock crew loud,

Figure 4.6: Some samples from synthetically generated images. 96 variations in standard fonts along with their normal, italic, bold and italic-bold variations are included in this database. Also, note that these images are degraded to reflect scanning artifacts.

described in Section 4.3. This database consists of three European languages, namely English, German, French, and a mixed set of all these languages. This dataset consists of 370,799 text-line images (number of text-lines in each language is given in Table 4.2). Some sample text-lines from this versatile database are shown in Figure 4.6.

- A large amount of text-lines used to measure the performance of script identification methodology (reported in Chapter 7), have been generated artificially. Different text corpora have been used to develop separate training and test data. There are 90,000 text-line images in the training set and 9,500 text-line images in the test set. The GT information is available in the form of both actual text and assigned class labels.

- To validate the hypothesis of the generalized OCR framework (reported in Chapter 8), a database from a English-Greek bilingual document has been created using the synthetic text-line image protocol. A total of 90,000 text-line images for the training phase and 9,900 text-line images for the evaluation purposes have been generated.

4.5 Chapter Summary

This chapter provides the benchmark databases for printed OCR of single as well as multiple scripts. There are two main approaches to develop such datasets, one is to use text and image alignment methods and the second is to generate text-line images synthetically using image degradation models. There are two main contributions of this chapter using these approaches. The first contribution of this chapter is a semi-automated methodology to develop ligature-level database for cursive scripts, although the same method can be used equally well to rapidly produce character-level database. The second contribution is to introduce various datasets based on the image degradation modelling approach. In this regards, various databases have been put forth, including a dataset for printed Devanagari OCR (Deva-DB), a subset of printed Polytonic Greek OCR (Polyton-DB) database, and three datasets for various MOCR tasks.

Chapter 5

OCR for Printed Modern Scripts

In recent times, Machine Learning (ML) based algorithms have been able to achieve very promising results on many pattern recognition tasks, such as speech, handwriting, activity and gesture recognition. However, they have not been thoroughly evaluated to recognize printed text. Printed OCR is similar to other sequence learning tasks like speech and handwriting recognition and therefore it can also reap the benefits of high performing ML algorithms.

Various challenges that are hampering the accomplishment of a robust OCR system have been discussed in Chapter 3. Upon looking at these challenges closely, one can realize that a human reader does not face many of these issues while reading a particular script. Human reading is powerful because of the ability to process the context of any text. Similarly, the internal feedback mechanism of Long Short-Term Memory (LSTM) networks enables them to process the context effectively; thereby rendering them highly suitable for text recognition tasks.

This chapter discusses the use of LSTM networks for the OCR on three modern scripts. The first part of the chapter, Section 5.1, overviews the complete design of the LSTM-based OCR system. The second part, from Section 5.2 to Section 5.4, reports the experimental evaluations for modern English, Devanagari and Urdu Nastaleeq scripts.

5.1 Design of LSTM-Based OCR System

This section provides necessary details about the LSTM-based OCR methodology that has been used for the experiments reported for modern English (Section 5.2), Devanagari (Section 5.3) and Urdu Nastaleeq (Section 5.4). The LSTM networks have been described in detail in Appendix A. For experiments reported in this chapter,

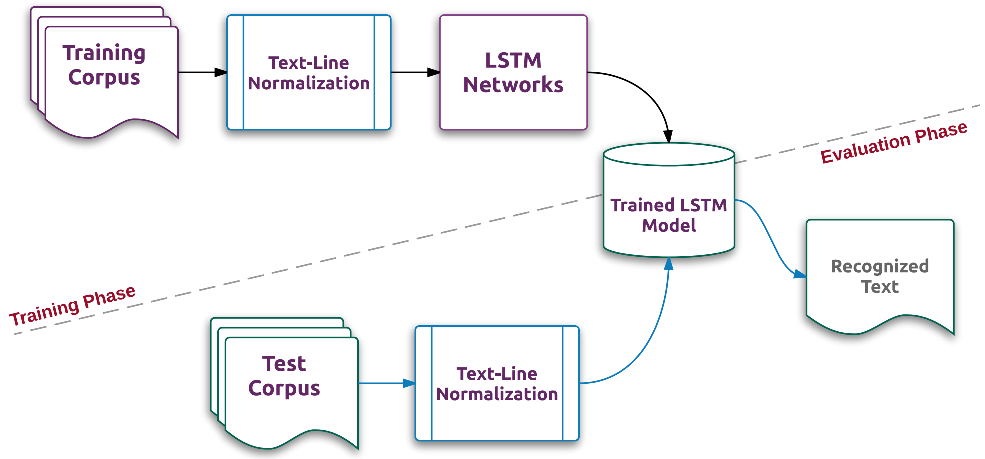

Training Corpus
Text-Line Normalization
LSTM Networks
Evaluation Phase
Trained LSTM Model
Recognized Text
Training Phase
Test Corpus
Text-Line Normalization

Figure 5.1: The complete pipeline for an LSTM-based OCR system. The text-lines are normalized before 1D-LSTMs can process them. During the training phase, the network learns the sequence-to-sequence matching in a supervised paradigm. At the test phase, the normalized text-line is given to the trained LSTM model, which generates the predicted sequence.

only 1D-LSTM networks have been utilized. MDLSTM architecture produced lower results in our preliminary experiments with printed English and Fraktur [BUHAAS13] and hence they are not considered. Some preliminary experiments on Urdu Nastaleeq script using Hierarchical Subsampling LSTM (HSLSTM) networks yield promising results; however, these networks have not yet been tested for other scripts.

The complete process of using LSTM-based OCR system is shown in Figure 5.1. To use 1D-LSTM networks, the text-line image normalization is the only important preprocessing step. This is due to the fact that these networks are not translation invariant in vertical dimension, so this dimension has to be fixed prior to using these networks. Various normalization methods that have been used in this thesis, are described in Appendix B.

There are few free parameters that are needed to be tuned in order to use 1D-LSTM networks and they are discussed in the following section. The features used for the LSTM networks are described in Section 5.1.2, while the performance metric is defined in Section 5.1.3.

5.1.1 Network Parameters Selection

There are four parameters, which require tuning in employing 1D-LSTM networks: the height of the text-line image, the hidden layer size, the learning rate, and the momentum. The input image height normalization depends upon the data. The specific values of these can be found in the relevant sections. However, momentum value is kept fixed at 0.9 for all the experiments. For the remaining two parameters (hidden layer size and learning rate), the suitable values are found empirically.

An analysis has been carried to find the optimal hidden layer size and learning rate for Urdu Nastaleeq script. In summary, best parameters for hidden layer size and learning rate are found to be 100 and 0.0001 respectively. These values also match those reported by other researchers [Gra12, GLF$^+$08] and they are also found to work well with other experiments reported in this thesis. Therefore, these values are kept fixed for all experiments in this chapter and for those reported in the rest of the thesis.

To find the optimal number of hidden layers, learning rate and momentum are kept constant at 0.0001 and 0.9 respectively. Various LSTM networks have been trained with hidden layer comprising of of 20, 40, 60, 80, 100, 120, 140 and 160 LSTM cells. The comparison of respective recognition errors on test set as a function of hidden layer sizes is shown in Figure 5.2. The training time as a function of hidden layer sizes is shown in Figure 5.3. From Figure 5.2 and Figure 5.3, we can deduce that, firstly, increasing the number of hidden layer size decreases the recognition error, but at the

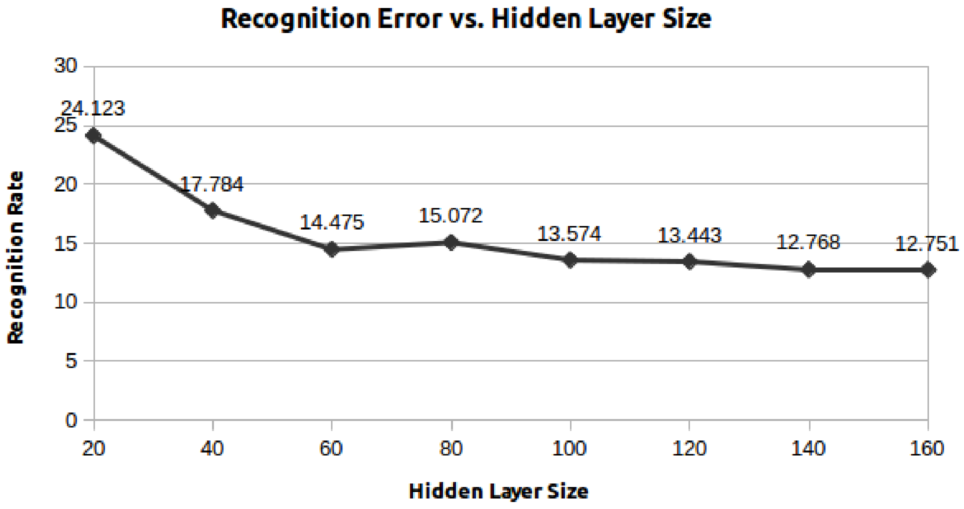


Figure 5.2: Recognition error versus hidden layer sizes. Recognition error decreases with the increase in the number of units in the hidden layer. However, it takes more time to train the network with increasing hidden layer size.

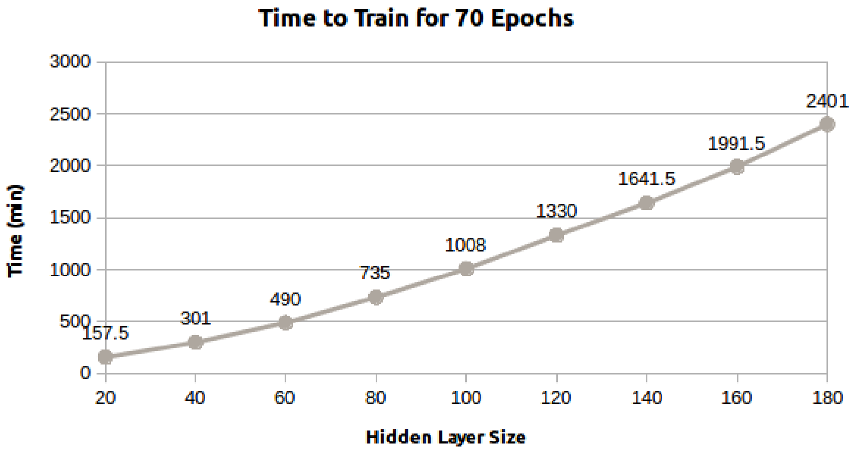


Figure 5.3: Training time as a function of hidden layer size. This time is computed for 70 epochs during the training on Intel Xeon 2.53 GHz, 40GB RAM desktop with Ubuntu 12.04 OS.

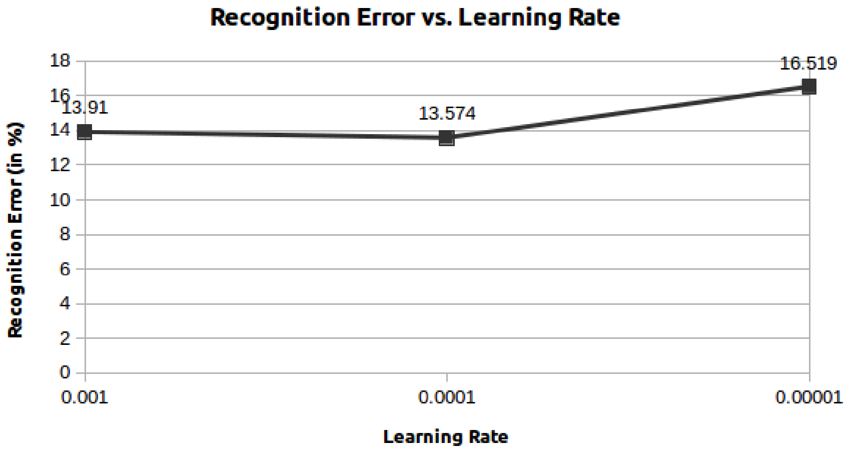


Figure 5.4: Recognition error versus various learning rates. Recognition rate is best observed when the learning rate is set to 0.0001.

same time, training for network with large size of hidden layer requires more time. Secondly, the increase in the training time is almost linear, while increase in the hidden layer size does not increase the recognition accuracy more than 5% when the hidden layer size is increased from 100 to 160. It is therefore decided to select 100 as the optimal hidden layer size for the present work.

In the next step, fixing the hidden layer size to 100, the learning rate is varied among 0.001, 0.0001 and 0.00001. The comparison of respective recognition errors on test set is shown in Figure 5.4. It is evident from the figure, that a learning rate of 0.0001 is the most suitable choice.

5.1.2 Features

The LSTM networks, like other modern machine learning methods, have shown to yield very good results on raw pixel values. No hand-crafted features are thus required and LSTM-based line recognizer learns the discriminating features from the input images itself. Yousefi et al. [YSBS15] demonstrated that 1D-LSTM networks with automatic feature learning can outperform MDLSTM networks and 1D-LSTM networks with hand-crafted features. Except the implicit features of baseline and x-height of individual characters, no other feature has been extracted for any work reported in this thesis.

5.1.3 Performance Metric

The recognition accuracy is measured as "Character Error Rate (CER) (%)", in terms of *Levenshtein Distance* (or more commonly known as the Edit Distance), which is the ratio between *insertion*, *deletion* and *substitution* errors and the total number of characters.

$$CharacterErrorRate(\text{CER})\ (\%) = \frac{I + D + S}{Total\ Characters} \times 100 \tag{5.1}$$

where '**I**' denotes the *Insertion*, '**D**' denotes the *Deletion* and '**S**' denotes the *Substitution* errors.

5.2 Printed English OCR

Printed English[1] has remained the main focus of the OCR research over many decades, and modern day OCR algorithm claim very low recognition errors. However, in practice, the generalization of such systems to new types of real-world documents is quite unsatisfactory. This happens mainly because of the nature of the OCR data, which means that the new text, on which an OCR system is applied, often differs significantly from all the training samples that that OCR system has seen during the training. There is currently no standard testing procedure or widely used dataset in the document analysis community to address this issue.

The following sub-section summarizes the research work reported so far for printed English. Experimental evaluation is discussed after describing the related work. LSTM-based OCR achieves very low error rates on standard database. An error analysis is presented at the end of this section.

---

[1] This section is based on the research work presented in [BUHAAS13]

5.2.1 Related Work

The OCR of printed English is very rich and an overview of decades of research in this field is prohibitively laborious. Mori et al. [MSY92] presented a historical review of OCR research and development, in 1992. They divided their review into two parts. In the first part, they described the history of OCR research, describing many *template matching* techniques and *structural analysis* methods. In the second part, they described various generations of commercial OCR products, starting from the early 1960's systems with very limited functionality to the commercial systems available between 1980 to 1990, such as hand-held scanners, flat-bed scanners and page readers.

The use of ANNs for text recognition began in the 1970s. Fukushima [Fuk80] presented a self-organizing neural network architecture to recognize simulated characters. This network consisted of two hidden layers and was trained in an unsupervised manner. The structure of this network was very similar to that proposed by Hubel and Wiesel [HW65] for visual nervous system.

LeCun [LC89] introduced Convolutional Neural Networks (CNN) for isolated handwritten numeral recognition. In this type of neural networks, a specialized layer, referred to as *feature maps*, scans the input image at different locations to extract features. Multiple feature maps were used to extract different features from the image.

Jackel et al. [JBB$^+$95] used the above-mentioned CNNs "*LeNet*" for handwritten character recognition. This system was based on segmentation-based philosophy, where individual character candidates were extracted at first from a given string. The *LeNet* was then applied to see if it can recognize this with high confidence or not. A simple threshold then identified the candidates that were recognized with higher probability. Others were discarded as incorrect segments.

Marinai et al. [MGS05] presented a survey on the use of ANNs for various tasks of a document analysis, including preprocessing, layout analysis, character segmentation, word recognition and signature verification. For the OCR task, they divided techniques for individual character recognition or word recognition.

Hidden Markov Models (HMM) were proposed by Rabiner [Rab89] in 1989 and they became extremely popular in the field of continuous speech recognition. Schwartz et al. [SLMZ96] adapted a speech recognition system for a language-independent OCR system, called “BBN BYBLOS”. On a multifont Arabic OCR corpus, they reported a character error rate of 1.9%. El-Mahallaway [EM08] proposed an Omni-Font OCR system for Arabic script using HMM methodology. However, HMM-based systems are mainly employed for the handwritten text recognition similar to automatic speech recognition.

Various attempts have been made to develop hybrid ANN/HMM methods to overcome the demerits of HMM-based methods. The ANN part of such systems extracts discriminative features and the HMM part is used as a segmentation-free recognizer. Rashid [Ras14] reported a system for printed OCR using this hybrid approach. In this method, Multilayer Perceptron (MLP) was used to extract the features from a given text-line. For this purpose, individual characters were extracted using the character segmentation method reported in [Bre01]. Features were extracted using a 30×20 window scanning over a character-segmented text-line. These features were extracted for the character and non-character classes. An AutoMLP [BS08] was used to learn them from 95 classes that include non-character (garbage) as well. HMM were then trained on these features using standard Baum-Welch algorithm. They reported an accuracy of 98.41% on standard dataset of printed English.

Most of the literature reported for these hybrid systems combine MLPs with HMM; however, Graves [GS05] described the use of RNN/HMM hybrid for phoneme recognition. They reported that using the RNN/HMM hybrid produces better results than those reported for ANN/HMM hybrid or simple HMM-based approaches.

5.2.2 Database

To evaluate the LSTM-based OCR method, we used UW3 dataset [LRHP97], representing 1,600 pages of document images from scientific journals and other common sources. Text-line images and corresponding GT text have been extracted from the data set using the transcriptions provided with this database. Text-lines containing mathematical equations are not used during either training or testing. Overall, we used a random subset of 95,338 text-lines in the training set and 1,020 text lines in the test set. Some of the sample images have been shown in Figure 3.1.

5.2.3 Experimental Evaluation and Results

In the context of these experiments, the text-lines are normalized to a height of 32 in a preprocessing step. Both left-to-right and right-to-left LSTM layers contain 100 LSTM memory blocks. The learning rate is set to $1e-4$, and the momentum to 0.9. The training is carried out for one million steps (roughly corresponding to 100 epochs, given the size of the training set). Test set errors are reported every 10, 000 training steps and plotted. The configuration of the trained LSTM network is shown in Figure 5.5. The two hidden layers correspond to the bidirectional mode of the LSTM network.

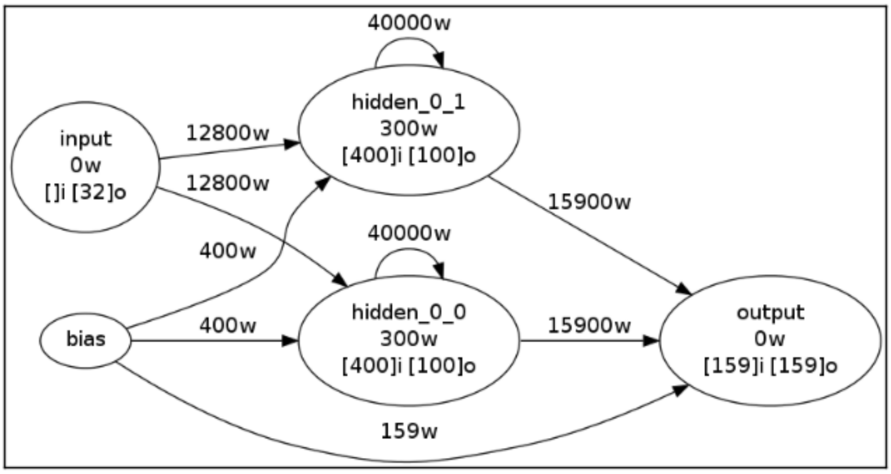


Figure 5.5: Configuration of trained LSTM network. There are 12,800 weights between the input and the hidden layers, while there are 15,900 weights between hidden layers and the output layer. There are 40,000 weights for self-looping states at the hidden layers. Input is a one dimensional vector with the depth of 32.

The process of LSTM training is illustrated in Figure 5.6. It is interesting to note from this figure that after seeing only 5400 text-lines, the network is able to recognize almost all characters. This shows that LSTM networks do not suffer generally from the 'over-training' problem. The LSTM network is able to achieve 0.6% (No. of total characters, N = 50,632) error on the test-set.

To compare the results with other contemporary OCR systems, the same test set has been utilized. The results have been compared with an old version of OCRopus [Bre08], Tesseract [Smi13] and ABBYY [ABB13] systems. The Tesseract system achieves a recognition error of 1.299% when run in line-wise mode with an English language model; ABBYY achieves 0.85% using the "English" setting, and the OCRopus achieved 2.14%. Figure 5.7 presents the comparison of the three OCR systems on the UW3 data collection. It should be noted that all these systems employ language modelling techniques to post-process the raw output, and in some cases other sophisticated techniques like font recognition and adaptivity. The LSTM network, on the other hand, achieves the results without any language modelling, post-processing, adaptation, or use of a dictionary. The running time is under a second for a regular text line on a modern desktop PC.

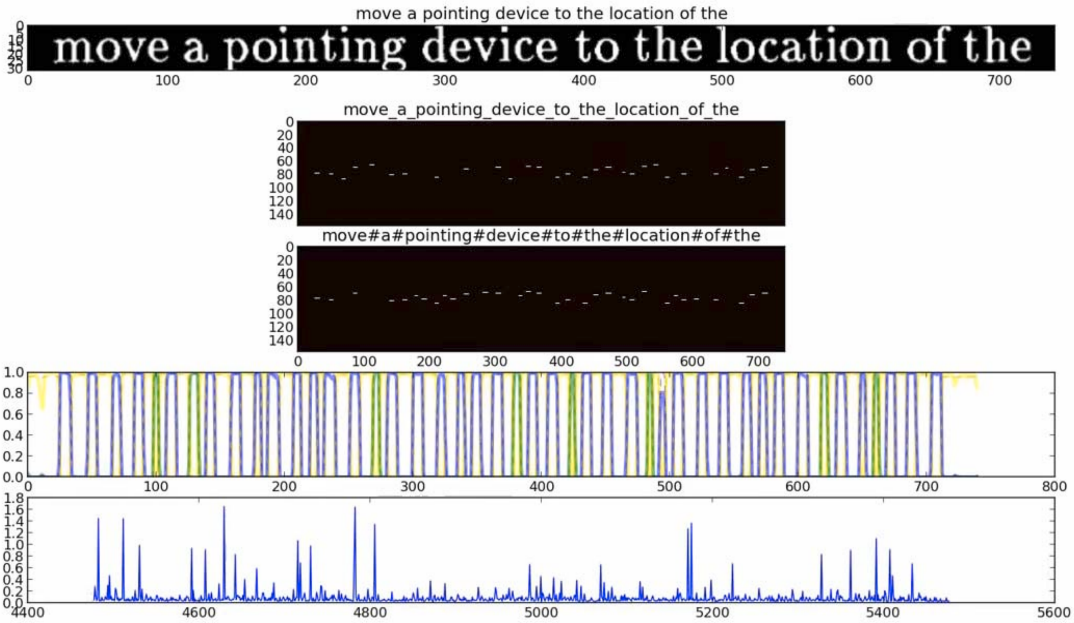


Figure 5.6: An illustration of the training steps of the LSTM line recognizer. The top row shows the input text-line along with the GT. The second row shows the *frame*-wise output from the LSTM network in image form, together with a decoded transcription. Frame-wise output from the LSTM network after using the CTC step is shown in the third row. The fourth row shows a graph of posterior probabilities for spaces (green), the top class (blue), and the reject class (yellow). The last row provides the graph of total error signal used in the back-propagation step. This error signal has a very non-Gaussian distribution, with occasional high spikes. These spikes seem to correspond to internal recognitions of the state-space used by the LSTM model.

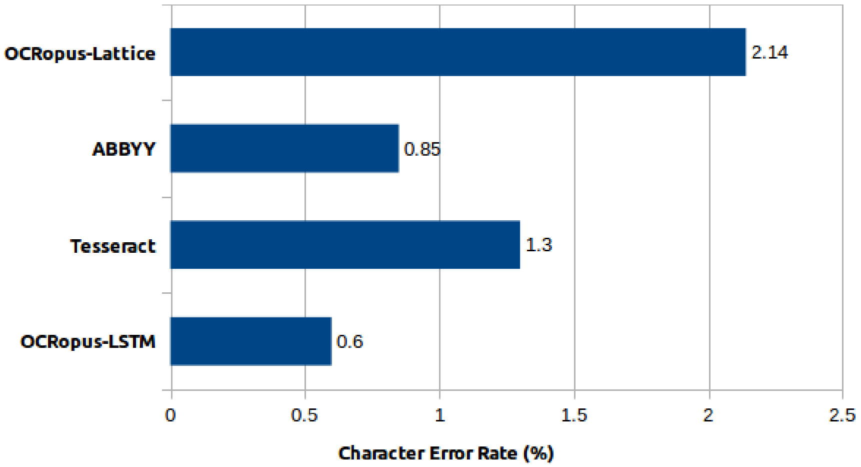
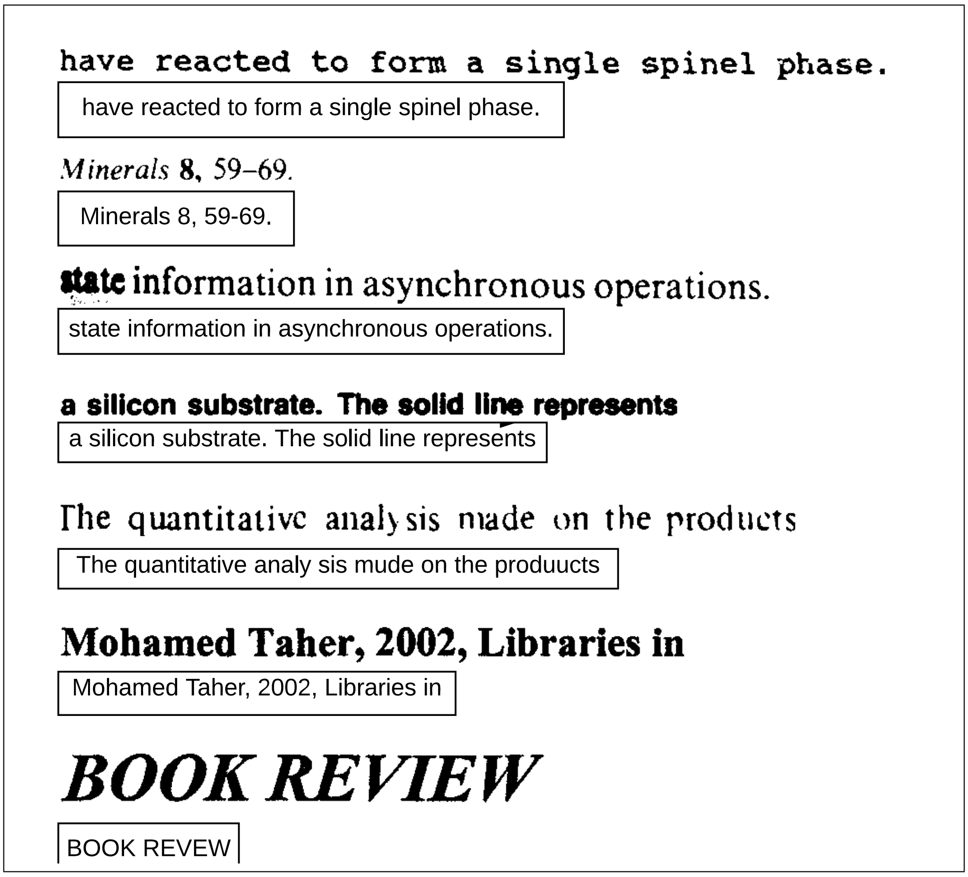


Figure 5.7: Comparison of CER of the LSTM recognizer (without a language model or adaptation) with the Tesseract [Smi13], ABBYY commercial OCR system [ABB13] and the OCRopus segmentation-based recognizer [Bre08]. Error rates on the UW3 database (English, modern print, N=8,988).

5.2.4 Error Analysis

There are a total of 313 errors and top confusions are ‘space’ deletions (34 times), ‘period’ confused with ‘comma’ (25), ‘space’ deletions (16), ‘period’ deletion (10), ‘comma’ confused with ‘period’ (6), ‘y’ confused with ‘v’ (5), ‘I’ deletions (5) and ‘i’ deletions (4).

Representative inputs and outputs are shown in Figure 5.8. The LSTM networks are able to recognize the text in a variety of font-sizes, styles and degradations (touching characters, partial characters). Errors appear when a significant part of a character is missing or in the case of capital characters.



have reacted to form a single spinel phase.

have reacted to form a single spinel phase.

Minerals 8, 59–69.

Minerals 8, 59-69.

state information in asynchronous operations.

state information in asynchronous operations.

a silicon substrate. The solid line represents

a silicon substrate. The solid line represents

The quantitative analysis made on the products

The quantitative analy sis mude on the produucts

Mohamed Taher, 2002, Libraries in

Mohamed Taher, 2002, Libraries in

BOOK REVIEW

BOOK REVEW

Figure 5.8: Input-output pairs from the LSTM line recognizer from modern English. Generally, LSTM copes well with different font styles, sizes and touching letters. Capital characters are more prone to misclassification, so are the uncommon names of people.

5.2.5 Conclusions

The results presented in this section show that the combination of text-line normalization and 1D-LSTM yields excellent results for English OCR. Improvements of 0.3% – 0.6% error may seem small, but they are enormously significant both in terms of differences between OCR systems and practical applications, greatly reducing the need for manual post-processing. These results suggest that error rates for LSTM-based OCR without any language model are considerably lower than those achieved by segmentation-based approaches, HMM-based approaches, or commercial systems, even with language models. Treating the input text line like a sequence of "frames" over time is related to HMM-based approaches [LSN$^+$99] and Document Image Decoding [KC94], but the LSTM approach has only three parameters, the input size, the number of memory units, and the learning rate.

5.3 Printed Devanagari OCR

Hindi, the national language of India, is based on the Devanagari script. It is the fourth most popular language in the world with 400 million speakers. A great wealth of ancient classical literature, scientific, and religious books are available in Hindi/Sanskrit. Therefore, there is a great demand to convert them into machine readable documents. Moreover, a well developed OCR technology in Hindi can assist in various fields as demonstrated by the OCR of Latin script.

However, despite the envisaged applications, the OCR research in Devanagari is still behind that of the Latin scripts. This can be attributed to various challenges. Section 3.5 discusses some of the challenges pertinent to Devanagari OCR in detail.

This section[2] describes the LSTM-based OCR solution for the printed Devanagari script. A short literature review is given in the following section before presenting the details about the experimental design and the results obtained.

5.3.1 Related Work

The first efforts towards the recognition of Devanagari characters in printed documents started in the 1970s. Researchers at Indian Institute of Technology, Kanpur, India developed a syntactic pattern analysis system for the Devanagari script [SM73]. In the 1990s, Chadhuri et al. [CP95] and Pal et al. [PC97] developed the first complete end to end OCR system for Devanagari. Although in the 1990s OCR for Devanagari was restricted only at research level, in the early 2000s it took a major leap when Center for Development of Advance Computing (CDAC) India released the first commercial Hindi OCR called "Chitrankan" [Pal04].

---

[2] This section is based on the work reported in [KUHB15].

Shaw et al. [SPS08] and Bhattacharya et al. [BPSB06] used HMMs for handwritten Devanagari character recognition. When using HMMs for OCR, statistical features play an important role. HMMs also require a large training set for estimating the parameters for a reliable recognition [Pal04].

Jawahar et al. [JKK03] used Support Vector Machines (SVM) for the recognition of printed Devanagari characters for a multilingual OCR engine. Principal Component Analysis (PCA) was used to reduce the dimensions of the feature space here. Even though SVMs are a good choice for the case where training data is limited, the main hurdle in use of the SVM method is the selection of a proper kernel [Pal04].

Bhattacharya et al. [BC09] used MLPs for the classification of Devanagari numerals. Each image was subjected to three MLP classifiers corresponding to a particular resolution. Singh et al. [SYVY10] used ANNs with features like mean distance, histogram of projection based on pixel location and their value. This ANN consisted of two hidden layers and was trained using the conventional backpropagation algorithm. ANNs are easier to implement and in addition to classification, also provide a confidence value for the classification [JDM00]. However, a major disadvantage with feed-forward ANNs is that they cannot remember the context.

Off late, LSTMs have also appeared in the Devanagari OCR research. Sankaran et al. [SJ12] has used bidirectional LSTMs for word classification in printed Devanagari documents with good results (Character Error Rate (CER) = 5.65%).

5.3.2 Database

A new database, *Deva-DB*, has been proposed in this thesis to advance the OCR research in the printed Devanagari. Further details about this database can be found in Section 4.4.1.

The test set is divided into two groups. The first set consists of 1000 synthetic text-line images generated from a different text corpus (other than the one used for training). The second set consists of a set of 621 real scanned text-line images (as described in Section 4.4.1) consisting of different fonts and different levels of degradations. Figure 5.9 shows few sample images from the second set containing the real scanned data.

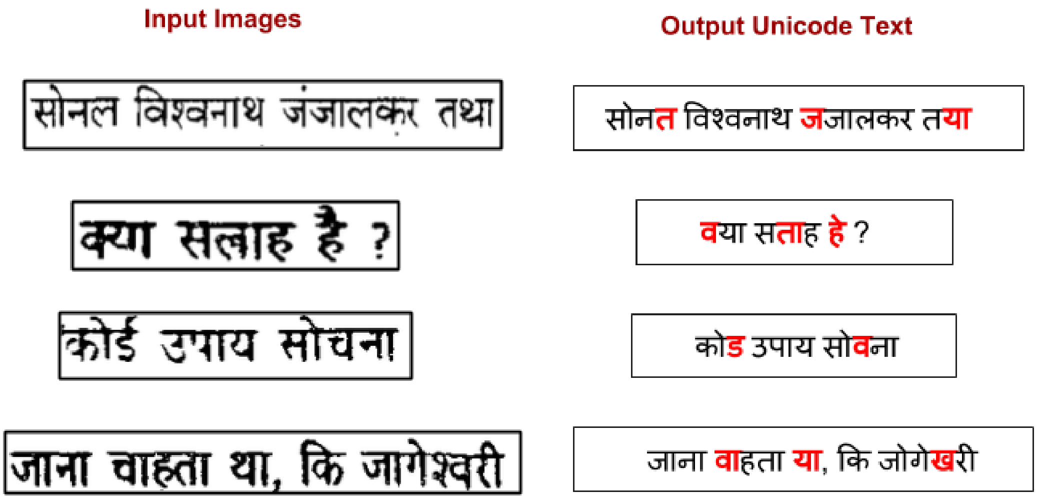


Figure 5.9: Samples taken from the real scanned set of line images where LSTM-based line recognizer fails. These samples not only vary in fonts but also in the degree of distortion due to ink spreads and distortion from the scanner. The corresponding Unicode output from the LSTM is also shown on the right side of each image. The errors in the output are marked in red. The errors have mostly occurred at places where the ink spreads are such that the LSTM could not differentiate between two similar characters.

5.3.3 Experimental Evaluation and Results

Three experiments, differing in the number of fonts for train and test data, have been performed. These experiments can be classified into (i) Single Font-Single Font (train-test), (ii) Multi Font-Single Font, and (iii) Multi Font-Multi Font. These experiments differ significantly from the experiments done by Sankaran et al. [SJ12] since we use the whole unsegmented line images (which includes a space character as a label in the output layer) for training the network. Moreover, no handcrafted features are extracted from the text-line images.

The line recognizer from the OCRopus OCR system has been adapted to work with Devanagari script. The LSTM network has been trained at two levels. The first one is with images only from a single font (*Lohit Hindi*) and the second one is trained with a training set consisting of images from multiple fonts (*Lohit Hindi, Mangal, Samanata, Kalimati, NAGARI SHREE, Sarai, DV ME Shree*). The results of the experiments are summarised in the Table 5.1.

Table 5.1: Performance comparison of four sets of experiments performed on Deva-DB. These experiments differ in types of fonts used in training/evaluation and also the type of test data, i.e., synthetic or scanned.

No.	Training Data	Test Data	CER(%)
1	Single Font, 24000 images (1,855,410 chars)	Single-Font, Synthetic 1000 images (80,50 chars)	1.2
2	Multi-Font, 24000 images (1,855,410 chars)	Single Font, Synthetic 621 images (22,517 chars)	4.1
3	Multi-Font, 24000 images (1,855,410 chars)	Multi-Font, Synthetic 621 images (22,517 chars)	2.7
4	Multi-Font, 24000 images (1,855,410 chars)	Multi-Font, Real Data 621 images (22,517 chars)	9.0

5.3.4 Error Analysis

Sample images of the scanned text-lines along with the LSTM-based line recognizer are shown in Fig. 5.9. Error made by the network is highlighted in red color. Apart from errors due to the confusions between similar shapes, one main reason for errors in these images is the *ink spread* on scanned text-line images. This ink spread makes it harder for LSTM networks to correctly recognize similar characters.

Although the confusion matrix in Table 5.2 shows most of the confusions to be similar in shape, the top confusions (ं, ्, र) happen to be deletions (network missing the character) or insertion (network erroneously inserting a character). This appears very strange at the first look. But a deeper analysis of the problem leads us to the conjunct characters in Devanagari.

The character 'ं' is a vowel which when combined with a consonant appears as a dot on the top of the consonant. For instance, when combined with the consonant 'त' (tha), the conjunct character would appear as 'तं' (than). A network trained with distorted images, can treat 'ं' (with a high probability) as a distortion in the image and would fail to detect it in some cases. The first image in Fig. 5.9 shows an example of such a deletion on the character ज.

The character '्' indicates that the preceding character should fuse with the succeeding character of '्'. If 'प' (pa) is to fuse with **character 'र'(ra)**, the code sequence would be 'प' + '्'+ 'र'. The compound character has a shape 'प्र'(pra) and it can be seen that the shape of the compound character represents 'प' (pa) more than 'र' (ra). This is the case with all the consonants which fuse with 'र' i.e. they take the shape of the first consonant.

Table 5.2: The top confusions from all four experiments (see Table 5.1) and from Tesseract OCR Engine. '_' corresponds to a deletion (Pred.) or an insertion (GT). The top two confusions are deletions or insertions in all but one case.

Experiment1			Experiment2			Experiment3			Experiment4			Tesseract		
Count	**Pred**	**GT**	**Count**	**Pred**	**GT**	**Count**	**Pred**	**GT**	**Count**	**Pred**	**GT**	**Count**	**Pred**	**GT**
521	_	्	103	्	_	119	_	ं	97	्	_	57	_	.
323	_	र	91	_	ं	108	व	ब	89	.	,	45	_	ं
317	_	ं	63	_	र	88	_	्	83	व	ब	40	व	ब
265	्	_	59	व	ब	66	_	र	74	_	ं	24	ॖ	_
235	ू	ृ	52	.	,	53	ं	_	65	_	र	23	ो	ी
221	_	.	51	ो	ी	50	प	म	56	ो	ी	18	जा	आ
206	त	न	42	य	थ	49	ी	ौ	41	प	म	17	ब	व
183	व	ब	42	त	न	49	.	,	40	_	,	12	।	!

Therefore, this is highly probable that the network might predict it as 'प' (pa). When this happens, we have a deletion of two characters which are '्' and 'र'. This explains why the top three confusions are deletions. In the fourth image in Figure 5.9, the conjunct character 'श्र' is replaced by consonant 'ख' because of a similar shape. This is one such example of deletion of '्'.

The top deletion errors can be reduced by taking into consideration the pixel variation in the vertical direction as well. Whereas other substitution errors can be removed by training on data which has more samples of these substituted shapes.

To compare the performance of LSTM networks, we evaluated the well known OCR engine Tesseract [Tes14] on our real test set. Tesseract results showed an error of 11.96% (with the default Hindi model) on the same test set where LSTM networks outputs an error of 9%. The confusion matrix from Tesseract also shows the top confusions to be deletions. The character 'ं' also appears as the top deleted character in the confusion matrix.

5.3.5 Conclusions

The complex nature of the Devanagari script (involving fused/conjunct characters) makes the OCR research a challenging task. Since words in Devanagari have a connected form, we proposed LSTM as a suitable classifier. A new database, *Deva-DB*, comprising of GT text-line images from various scanned pages and synthetically generated text-lines, has been proposed. OCRopus line-recognizer has been adapted and trained on this database. This LSTM-based system yields a character error rate of 1.2% when the test fonts matched that of the training data but the error rate increased to 9% when tested on scanned data (containing different set of fonts). The important issue that the network faced while classifying the characters is that of conjunct characters and the cases where characters are vertically stacked. The shape and position of these vertically stacked glyphs vary widely with different fonts.

5.4 Printed Urdu Nastaleeq OCR

Urdu, the national language of Pakistan, is written in the Nastaleeq script. This script belongs to the Nabataean family of scripts and shares many characteristics with Arabic and Persian. Some of its salient features that pose specific challenges for its OCR are presented in Section 3.6. Nastaleeq script[3] consists of 45 basic characters. Five

---

[3] A part of this section is based on the work reported in [UHASB13].

Category	Urdu Characters	Classes
Characters that can occur only in isolation (CAT-I)	آ، ژ، ۓ، ؤ، ء	5
Characters that can not occur in the beginning or in the middle of a ligature (CAT-II)	ا، د، ڈ، ذ، ر، ڑ، ز، ں، و، ے	20
Characters that may occur in isolation, at the beginning, at the end, or in the middle of a ligature (CAT-III)	ب، پ، ت، ٹ، ث، ج، چ، ح، خ، س، ش، ص، ض، ط، ظ، ع، غ، ف، ق، ک، گ، ل، م، ن، ہ، ھ، ی	108
Characters that can occur only in isolation or in the middle of a ligature (CAT-IV)	ئ	2
Characters that can occur only at the end of a ligature (CAT-V)	ۃ، ۂ	2
Honorific marks (CAT-VI)	ؑ، ؓ، ؒ، ؐ، ﷲ، ﷺ، ، ، ﷽، ﷻ	8
Punctuation marks (CAT-VII)	/، ٪، ٖ، ، !, >,), (، <،]، [، }، {، ؟، ۔، :، ”، '؛ ٰ، ّ، َ، ِ، ُ، ً	26
Numerals (CAT-VIII)	۰۱۲۳۴۵۶۷۸۹، 0123456789	20

Figure 5.10: Categorization of Urdu characters. There are a total of 45 characters in eight categories based on their position in a ligature.

(05) characters can occur only in isolation, 10 can only occur in the first position or at the last position, 2 characters can occur only at the end of a ligatures, and only 1 character can occupy position in the middle; it cannot be located in any other position. Remaining 27 characters may occur in isolation, at the beginning, at the end or in the middle of a ligature. Moreover, there are 26 punctuation marks, 8 honorific marks, and 20 digits. Some common punctuations (like %, <, >, parentheses, etc.) and English numerals are also used in Urdu publications frequently; so they are also included in the list of possible characters/class-labels. Characters belonging to above-mentioned eight categories are shown in Figure 5.10. So, in total there are 99 individual labels. Moreover, if we take the shapes of various characters as a separate label, then there are 191 labels. The last column in Figure 5.10 details the number of classes in each category as per their number of shapes depending on their position in a ligature.

Various experiments have been performed on Urdu Nastaleeq script using the LSTM networks. The experimental setup and the results obtained by applying two different LSTM architectures are described in the following sections. However, a short overview of work reported for Nastaleeq OCR is given first to introduce the little work that has been carried out so far in this field.

5.4.1 Related Work

Urdu Nastaleeq script has not yet been thoroughly introduced to current OCR research and till date no significant OCR system, specialized for Urdu Nastaleeq script has been reported. However, there has been an increase in the interest of the researchers, especially from the Indian subcontinent to address the Urdu Nastaleeq OCR problem. Some earlier work was done by Pal et al. [PS05] to recognize individual Urdu characters. They recognized these characters using a combination of topological, contour and water-reservoir concepts. Segmentation of Urdu ligatures is very complex and error-prone. As a result, there has been a shift in interest among researchers to try segmentation-free approaches.

Sabbour et al. [SS13] presented a segmentation-free approach for Urdu and Arabic scripts. They modified the traditional shape-context features method [BMP02] to extract features from Urdu/Arabic ligatures and then applied k-nearest neighbour classifier to recognize the ligatures. Instead of computing shape-context features of the whole ligature, they first divided the ligature image into four parts; they then computed the shape-context of points in each of the region separately. Subsequently, these features were concatenated to define the cumulative feature vector.

Naz et al. [NHR+13] presented a comprehensive survey of Urdu document image analysis research (both from layout analysis and OCR points of view). In another work, Naz et al. [NHR+14] outlined various difficulties in detecting baseline for Nastaleeq script.

Akram et al. [AHN+14] recently adapted the well-known Tesseract OCR system for Urdu Nastaleeq. They modified various routines in the Tesseract system to work for cursive script like Nastaleeq. They reported an accuracy of 97.87% and 97.71% for 14- and 16-point font-sizes respectively.

5.4.2 Database

Urdu Printed Text Images (UPTI) dataset [SS13] has been used for the evaluation of the LSTM-base line recognizer. This Urdu dataset consists of 10,063 synthetically generated text lines. Various degradation techniques [Bai92] have been applied to increase the size of the dataset. 12 sets were generated by varying four parameters, namely, *elastic elongation*, *jitter*, *sensitivity* and *threshold*. This dataset contains both ligatures and text-line level GT; however, only text-line dataset is used for the present work. These lines are divided into three sub-categories, training (46%), validation (34%) and testing (20%). Each set is build such that text-lines from all 12 degraded

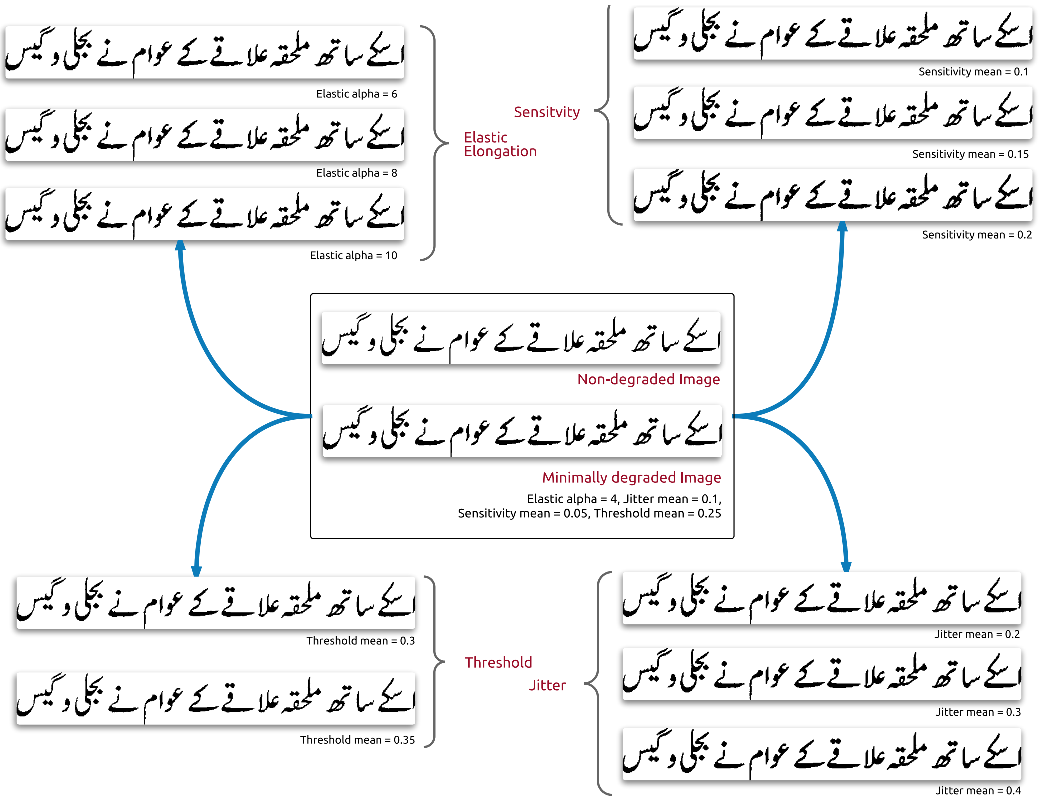

Elastic alpha = 6
Elastic alpha = 8
Elastic alpha = 10
Elastic Elongation
Sensitvity
Sensitivity mean = 0.1
Sensitivity mean = 0.15
Sensitivity mean = 0.2
Non-degraded Image
Minimally degraded Image
Elastic alpha = 4, Jitter mean = 0.1,
Sensitivity mean = 0.05, Threshold mean = 0.25
Threshold mean = 0.3
Threshold mean = 0.35
Threshold
Jitter
Jitter mean = 0.2
Jitter mean = 0.3
Jitter mean = 0.4

Figure 5.11: Sample text-line images from the UPTI database. A single text-line rendered with different values of degradation parameters is shown.

categories and 1 clean category are taken in equal proportions. It is also possible to take all images from all the categories (resulting in 13 × 10,063 text-line images) to increase the size of the database; however, it is not done in this work because it would have make LSTM learn the training data too well, thereby resulting in poor generalization. Some of the text-line from UPTI database are shown in Figure 5.11.

5.4.3 1D-LSTM Networks for Urdu Nastaleeq Script

There are two kinds of experiments performed for Urdu Nastaleeq script using single layer 1D-LSTM networks. In the first evaluation, shape variations at all four positions (isolation, beginning, middle and end) are considered (191 classes). In the second evaluation, only basic labels are considered (99 classes).

Since 1D-LSTM networks are not translation invariant in vertical dimension, it is therefore necessary to normalize the input images to a specific height, so that the depth of 1D frame is consistent over all text-line images. Currently, there are no Nastaleeq-specific normalization method reported. In the current work, each text-line image is rescaled to a fixed height. Raw pixel values are used as features and no other sophisticated features are extracted. A 30 × 1 window is traversed over the

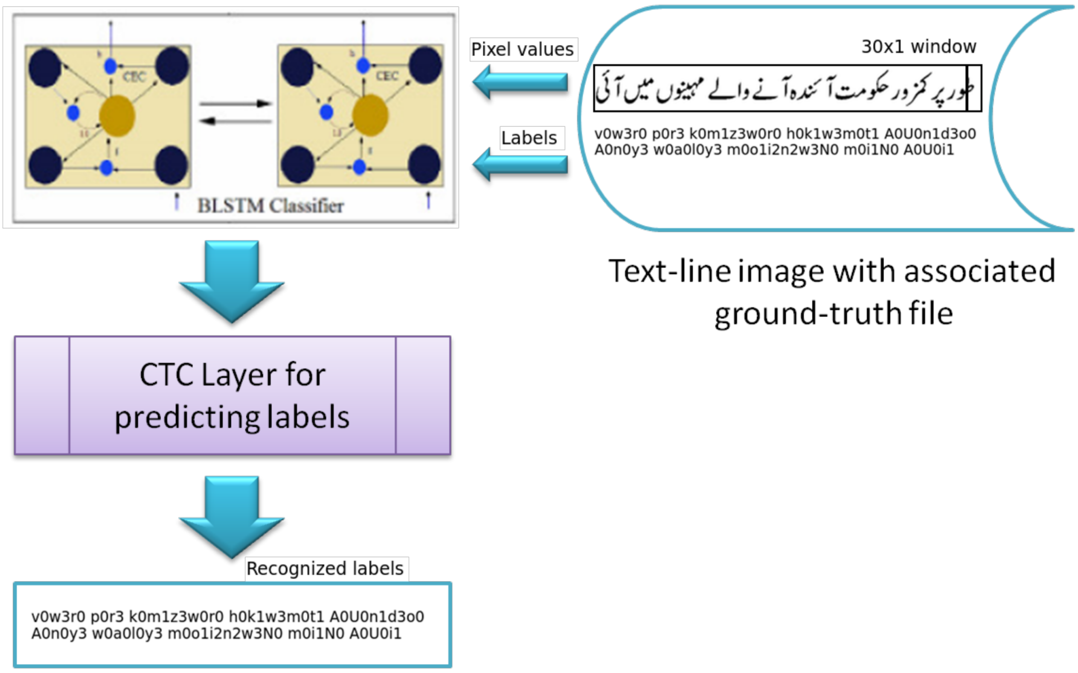

CEC
CEC
BLSTM Classifier
Pixel values
Labels
30x1 window
طور پر کمزور حکومت آئندہ آنے والے مہینوں میں آئی
v0w3r0 p0r3 k0m1z3w0r0 h0k1w3m0t1 A0U0n1d3o0 A0n0y3 w0a0l0y3 m0o1i2n2w3N0 m0i1N0 A0U0i1
Text-line image with associated ground-truth file
CTC Layer for predicting labels
Recognized labels
v0w3r0 p0r3 k0m1z3w0r0 h0k1w3m0t1 A0U0n1d3o0 A0n0y3 w0a0l0y3 m0o1i2n2w3N0 m0i1N0 A0U0i1

Figure 5.12: Training pipeline for Urdu Nastaleeq OCR. A 30 × 1 window traverses the height-normalized image and the LSTM network is provided with a 1D sequence of corresponding pixel values. The CTC layer then performs the output-transcription alignment.

text-line image and the resulting 1D sequence (frame) is fed to the LSTM network for training.

As mentioned earlier, the LSTM architecture with CTC output layer is employed to evaluate RNN for Urdu script. A publicly available RNN library [Gra] is used for evaluation. Implementation of both 1D and multidimensional LSTM networks is provided in this library along with CTC output layer.

Size of the hidden layer, learning rate and momentum are other tunable parameters. For the training purpose, the normalized gray-scale input text-line image is scanned from left to right to extract the features. The corresponding transcriptions are reversed to make it consistent with the input image (Urdu is read from right to left). Figure 5.12 shows the complete training pipeline.

Normalized text-line images along with their transcriptions are fed to the network, which perform the forward propagation step first. Alignment of output with associated transcriptions is done in the next step and then finally backward propagation step is performed. After each epoch, training and validation error are computed and the best results are saved. When there is no significant change in training and validation errors for a preset number of epochs, the training is stopped. Training and validation errors are recorded and the network is evaluated on test set.

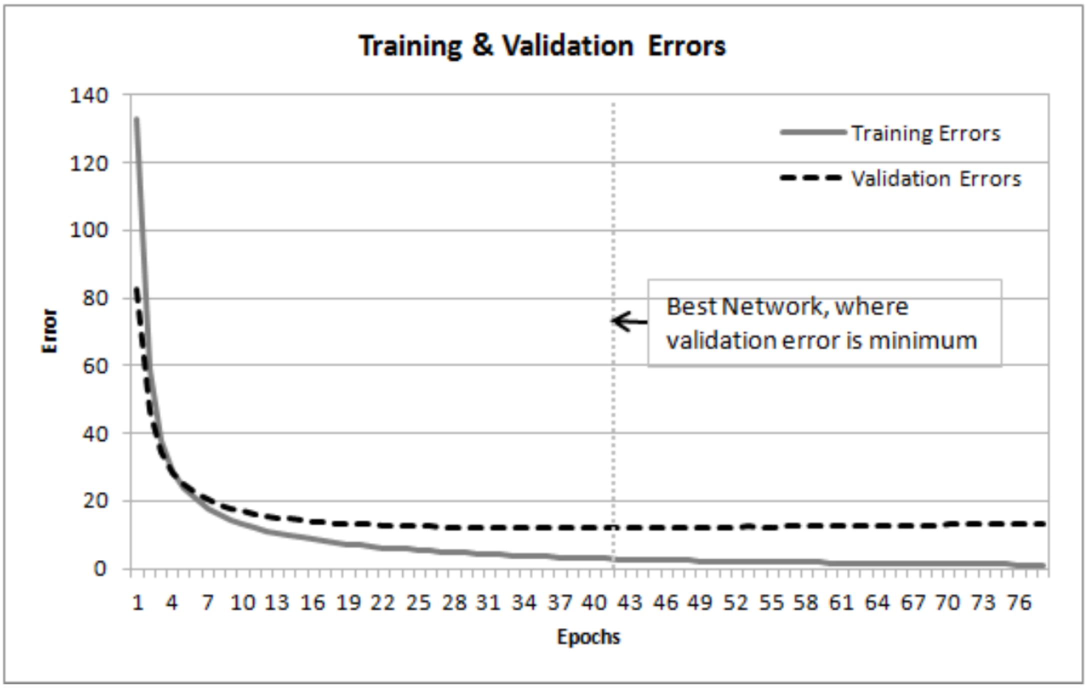
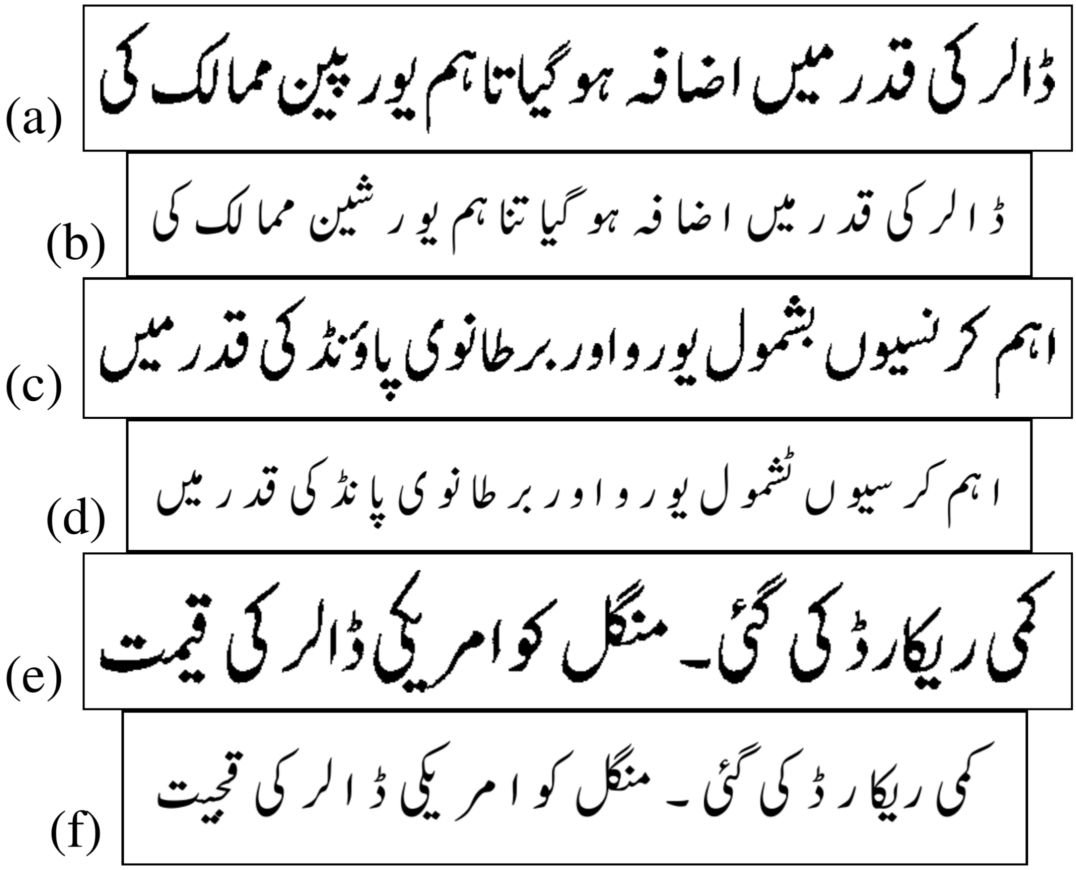


Figure 5.13: CER during the training phase. The validation error is minimum at 41^{st} epoch.

Parameter tuning is discussed in detail in Section 5.1.1. Briefly, best value for hidden layer size and the learning rate are 100 and 0.0001 respectively. For this network with best parameters, training and validation errors as a function of number of epochs are shown in Figure 5.13. This network took 77 epochs to converge. However, it can be seen that the validation error is minimum after 41 epochs (marked as a dotted-line in Figure 5.13). This network is returned as the best network.

To evaluate the generalization capability of the trained LSTM models, a sub-set of UPTI dataset is used as a test set. There are 2,003 randomly selected text-line images in this test set. As mentioned before, LSTM networks have been evaluated for two scenarios: considering ligature shape variations and ignoring shape variations in the ligatures. For the first case, the recognition error is 13.574%, (Total No. of labels, N = 74, 279), while for the second case, the recognition error is 5.15% (N = 74, 279).

There have not been many OCR systems available for Urdu Nastaleeq script. Only shape-matching based OCR system proposed by Sabbour et al. [SS13] is reported in recent times. They evaluated their system on clean printed text as well on some of the artificially degraded versions of the clean dataset. They achieved 11.2% letter error rate on clean images. They also reported error rates for various degradation effects on individual basis. There is no error rate reported for mixed dataset that we used in our evaluations. Moreover, they did not consider the case where ligature shape variations are not considered (where we achieved a CER of 5%). It is therefore



(a) ڈالر کی قدر میں اضافہ ہوگیا تاہم یورپین ممالک کی

(b) ڈالر کی قدر میں اضافہ ہوگیا تاہم یورشین ممالک کی

(c) اہم کرنسیوں بشمول یورو اور برطانوی پاؤنڈ کی قدر میں

(d) اہم کرسیوں ٹشمول یورو اور برطانوی پانڈ کی قدر میں

(e) کمی ریکارڈ کی گئی۔ منگل کو امریکی ڈالر کی قیمت

(f) کمی ریکارڈ کی گئی۔ منگل کو امریکی ڈالر کی قجیت

Figure 5.14: Input/output from the LSTM-based OCR illustrating capabilities and errors. (a), (c) and (e) represent the original images, whereas (b), (d) and (f) represent the output of LSTM network.

not possible to do one to one comparison in true sense, but it can be seen that our system performed better taking into consideration that clean images are only $\frac{1}{13}$ of our test-dataset. Secondly, the performance of their system changes significantly by changing degradation parameters' values.

5.4.4 Error Analysis

Some sample input-output pairs are shown in Figure 5.14. LSTM network performs generally well for most of the labels; however, it appears that it sometimes fails to recognize the location of dots and diacritics (e.g. Figure 5.14-(a)-(e)). As mentioned earlier, the dots and diacritics are very important to give meaning to a ligature. Other errors are mostly due to very similar shapes of a ligature (e.g. Figure 5.14-(e) and (f)).

The results of two evaluations are surprising, because, at the beginning of experiments, it was perceived that incorporating the shape variations as separate classes would increase the recognition accuracy because we have less variations within a specific class. There could be two issues: by ignoring the shape variations, number of samples per class increase. So, increased numbers per class is resulting in better training and thereby reducing recognition errors. Secondly, when considering shape variations, number of samples per class are small and that could lead to insufficient training and thus resulting in higher recognition errors. One may argue that less number of classes generally means better classification accuracy; however, it should be noted that the dataset remains the same, so by merging many classes, we actually are increasing the variations. This conflict may be solved by having such a dataset in which samples per class in both variations are equivalent.

5.4.5 HSLSTM Networks for Urdu Nastaleeq OCR

The Hierarchical Subsampling LSTM (HSLSTM) networks are described in Appendix A. One advantage of these networks is to compress the sequence into blocks or windows, thus speeding up the training time with complex architectures. To use these networks with 1D frames (extracted by traversing a window whose width is 1-pixel), these networks combine more than one frame in a single timestep.

There are more free parameters in HSLSTMs than single layer 1D-LSTM networks owing to their complex architecture. The **number of hidden layers** is fixed to three as suggested in [Gra12]. The **number of LSTM cells** in each hidden layer is found empirically depending upon the character error rate. The **number of *tanh* cells in feedforward layers** are also found empirically. The **subsampling size** is set to '2' for both 1st and 2nd hidden layers.

Another important difference in HSLSTM networks from single layer 1D-LSTM networks is the use of a different text-line normalization method. Image rescaling does not yield good results when used with HSLSTM networks. The *Filter-based* text-line normalization method described in Appendix B is employed in this work. This method is applicable to many scripts and found to work good for Urdu Nastaleeq script. To find the right value of the text-line height, experiments are performed using both the 1D-LSTM and HSLSTM networks. The character error rate is plotted (see Figure 5.15) against various normalization heights and the best value is selected.

After having selected the normalization height to 85, the number of LSTM cells in each of the three hidden layers and the number of *tanh* units in feedforward layers are searched. Table 5.3 shows the character error rates for various configurations.

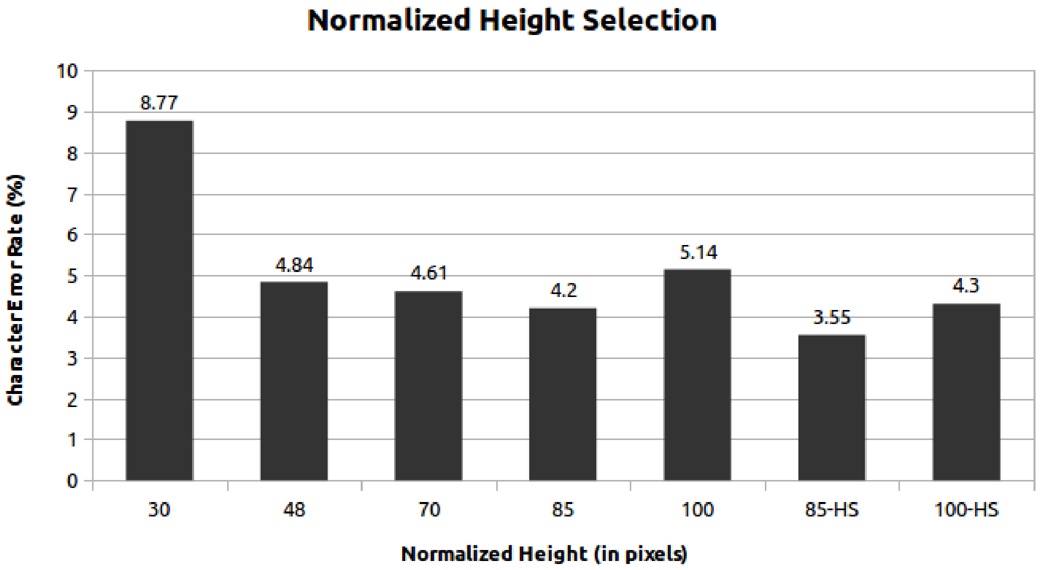


Figure 5.15: Comparison of CER for various values of normalized height. It is evident that the HSLSTM network with a normalized height of 85 (denoted by 85-HS) is the optimal choice. The number of LSTM cells in three hidden layers, from lower to higher, are 20, 60, and 100 respectively, while the number of *tanh* units in 2 feedforward layers are 20 and 60 respectively.

Table 5.3: Various architectures of HSLSTM networks are compared. It is noted that the bigger architectures tend to perform better. It must be kept in mind that the sample size is fixed to 2 in all these experiments.

Hidden Layer Size	Feedfoward Layer Size	CER (%)
2,10,50	6,20	6.12
2,10,50	12,40	5.47
4,20,100	6,20	4.07
4,20,100	12,40	3.32
20,60,100	6,20	3.01
20,60,100	12,40	2.55

5.4.6 Comparative Analysis

The UPTI database is becoming popular to benchmark the OCR algorithm for Urdu Nastaleeq script (see Table 5.4). Recently, Naz et al. [NUA^{+}15] reported the results of MDLSTM networks with hierarchical subsampling on this database. They trained MDLSTM networks with many hand-crafted features, which achieved slightly better error rates than those obtained using 1D-LSTM networks with single layer on the same test data (5.03% as compared to 5.15%). However, HSLSTM networks based on 1D architecture significantly outperform both of these results and achieve 2.55% character

Table 5.4: The comparison of various LSTM architectures for Urdu Nastaleeq OCR. The evaluations are performed on the same test data.

LSTM Architecture	Features	CER (%)
1D-LSTM	Automatic	5.15
MDLSTM [NUA$^+$15]	Hand-crafted	5.03
HSLSTM	Automatic	**2.55**

error rate. MDLSTM networks take huge amount of time to train and the results are still marginally better than those obtained by 1D-LSTM networks. On the other hand, HSLSTM networks training time is comparable to that of simple LSTM networks (24 compared to 17 hours on the same training data).

5.4.7 Conclusion

Urdu Nastaleeq is a challenging script for automatic recognition. Moreover, the research in this field is still in its infancy. LSTM networks seem to be a good choice to OCR this script, due to their excellent contextual processing capacity. Two architectures have been evaluated for printed Urdu Nastaleeq OCR. The HSLSTM network yields significantly better results than the simple 1D-LSTM networks. The training time for these networks is considerably small as compared to more complex MDLSTM networks based on hierarchical subsampling [GS08] that are reported in the literature for Arabic handwriting recognition.

5.5 Chapter Summary

This chapter discusses the application of LSTM-based OCR methodology for three modern scripts, namely English, Devanagari and Urdu Nastaleeq scripts. LSTM-based OCR solution has achieved very low error rates on printed OCR of these scripts. Results on English and Urdu Nastaleeq are the best reported OCR results among the reported results in the literature. Training of 1D-LSTM models for OCR is considerably simpler than previous training methods, since all the training is carried out in terms of text line images and transcriptions. In terms of parameters, beyond text line normalization, all that is required is choosing the input size and the number of internal state units. Moreover, the reported results are obtained without using any hand-crafted features, font recognition and adaptation, language modeling or dictionary correction. The results on Urdu Nastaleeq using HSLSTM networks are significantly better than those obtained using simple LSTM networks.

Chapter 6

OCR for Historical Documents

Preserving the literary heritage is important to gain valuable insights into human history and to gain knowledge about the different aspects of our ancestors' lives. Documents, whether written on leaves, stones, textiles, animal skin, or paper, have remained the eyes to the history of mankind. Old documents are very delicate and they need extreme care. With the advancement in the capturing technology, the cost of preserving old documents has decreased many folds. Libraries and institutes around the globe have made valuable efforts in making digital copies of historical documents.

However, navigating through these documents is still very difficult as the scanned images are not suitable for searching and indexing. Automatic recognition of historical documents can help a Paleographer[1] by indexing the old documents in a digital library. But, one must contemplate that this is very challenging due to the following reasons:

- These documents are usually found in a very bad condition, so the quality of text is not very high, and is often highly degraded with torn pages.
- The document may have been written in an ancient script with archaic orthography. These primitive scripts render these documents very difficult to recognize by computer programs.
- Many modern text recognition algorithms, based on supervised ML, require a lot of transcribed training data. Transcribing documents by a human is very laborious and costly due to the involvement of language experts; therefore, it is not feasible to transcribe a large enough data for historical scripts.

---

[1] Paleography is the study and scholarly interpretation of ancient writings and its various forms.

This chapter contributes in the following two ways to enhance the research in the field of historical document digitization.

- Firstly, LSTM-based OCR methodology, which has been described in the previous chapter, is successfully applied on Frakur and Polytonic Greek scripts. On both these scripts, the LSTM networks have been trained using the synthetically generated data and the results on scanned documents have shown that the LSTM-based methodology has outperformed both Tesseract and ABBYY OCR systems.
- Secondly, a novel approach has been proposed to OCR scripts that lack large amounts of training data. This approach combines the segmentation-based and segmentation-free OCR paradigms to achieve this goal. Although meagre GT data was available, the proposed approach has shown to produce excellent OCR results without using any language models or any other technique.

The first part of this chapter, Section 6.1 reports the results of applying LSTM-based OCR to Fraktur and Polytonic Greek scripts. The second half, Section 6.2 describes a novel framework to combine segmentation-based and segmentation-free approaches for 15^{th} century Latin script.

6.1 Fraktur and Polytonic Greek Scripts

Fraktur script, also knows as Black-Letter or Gothic, has been the script of writing in central Europe, especially in Germany from 16^{th} to 20^{th} century. Polytonic Greek , on the other hand, remained the main script in Greece till the 1980s. Rich literature heritage of both countries is available in these scripts, and OCR research is actively being utilized under various digitizing projects. This section describes the results of experimental evaluation of LSTM-based OCR methodology (see Section 5.1) for Fraktur and Polytonic Greek scripts.

6.1.1 Related Work

There is not much literature available regarding the digitization of Fraktur and Polytonic Greek scripts. ABBYY[2] is providing support for Fraktur OCR for a long time. Its core OCR module is based on segmentation-based approach [Fuc] where a combination of classifiers recognize a single letter. Furrer and Volk [FV11] improved both the character and word error rates using language modeling. Tesseract [Tes14] also provides the OCR capability for Fraktur script.

---

[2] `http://www.frakturschrift.com/en:start`

White [Whi13] adapted Tessereact OCR system and proposed many improvements to better train it for Polytonic Greek script recently. They showed how to incorporate language-related hints to improve the performance. Boschetti et al. [BRB+09] compared three OCR engines including ABBYY FineReader 9.0, OCRopus 0.3 (based on Tesseract) and Anagnostis 4.1 for Polytonic Greek documents. They aligned the outputs of three OCR systems using progressive multiple alignment method to improve their results.

Gatos et al. [GLS11] proposed a Polytonic Greek recognition system consisting of five modules. One module was dedicated to accent recognition, while the remaining four modules recognized character belonging to various horizontal zones. At the first step, accents were identified and separated. The remaining characters were segmented using skeleton features of foreground and background pixels. The segmented characters were assigned to one of four zones identified by the baseline and the mean-line. In this manner, characters were divided into four categories. A normalization step is carried out on all five categories (modules) before using k-NN classifier, with k=3. They reported a character recognition accuracy of 90% on various Polytonic Greek documents that belonged to the period between 1950 to 1965.

6.1.2 Database

Both Fraktur and Polytonic Greek lack any standardized datasets to evaluate the performance of OCR algorithms. For the OCR of Fraktur script, the LSTM training is carried out on synthetically generated 20,000 text-lines collected from various Fraktur sources. And for ancient Greek, the training is done using a combination of scanned and artificial data from *Polyton-DB* (see Section 4.4.2).

To evaluate the performance of LSTM models on Fraktur, some sample images are taken from two scanned books, namely (i) Theodor Fontane Wanderungen durch die Mark Brandenburg (a clean, high resolution scan), and (ii) the Ersch-Gruber encyclopedia (a noisy, lower resolution scan).

The LSTM-base OCR models for old Greek are evaluated on various combinations of datasets. These combinations are listed in Section 6.1.4.

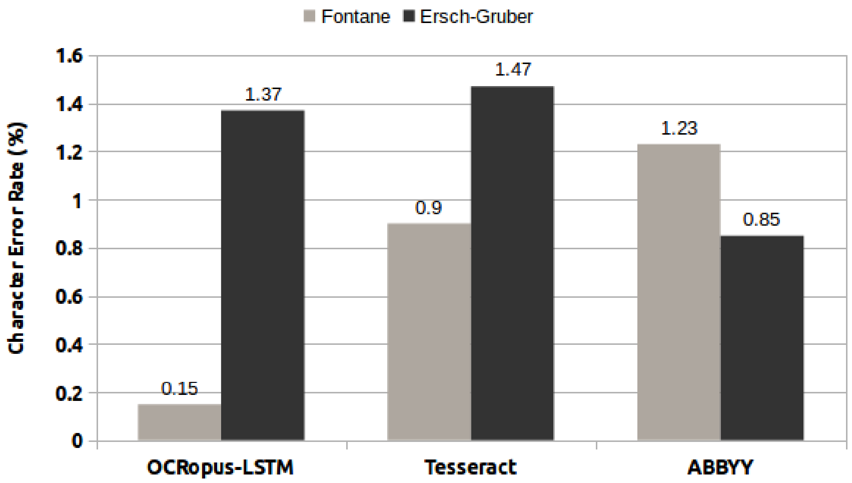


Figure 6.1: Comparison of CER of the LSTM recognizer (without a language model or adaptation), the Tesseract [Smi13] and ABBYY commercial OCR system [ABB13]. Error rates on the Fontane dataset (German, Fraktur, N=8,988) and the Ersch-Gruber dataset (German, Fraktur, N=10,881).

6.1.3 Experimental Evaluation for Fraktur Script

Tests have been performed on two scanned documents of Fraktur script. Furthermore, text in Antiqua (Latin) font was excluded from the evaluation. Since error rates were so low, we could quickly determine error rates and Ground-Truth (GT) data using a spell checker and verify any flagged words against the source image. The text contains few digits and little punctuation marks which yields good error estimates. On randomly selected pages from Fontane representing 8,988 Fraktur characters, the CER is 0.15%. On Ersch-Gruber, the CER is 1.37% on randomly selected page images representing 10,881 Fraktur characters. These results are without a language model and without adaptation to the fonts found in these documents. Recognition results for Fraktur (for both Fontane and Ersch-Gruber) are also compared with other OCR systems (see Figure 6.1). The Tesseract system applied to these inputs yields CER of 0.898% (Fontane) and 1.47% (Ersch-Gruber), using a German dictionary and font adaptations. ABBYY commercial OCR system outputs CER of 1.23% on Fontane and 0.85% on Ersch-Gruber.

6.1.4 Experimental Evaluation for Polytonic Greek Script

In order to evaluate the LSTM-based line recognizer, three types of experiments have been carried out using different combinations of documents in Polyton-DB. Like Fraktur, the results have also been compared with the Tesseract and ABBYY FineReader.

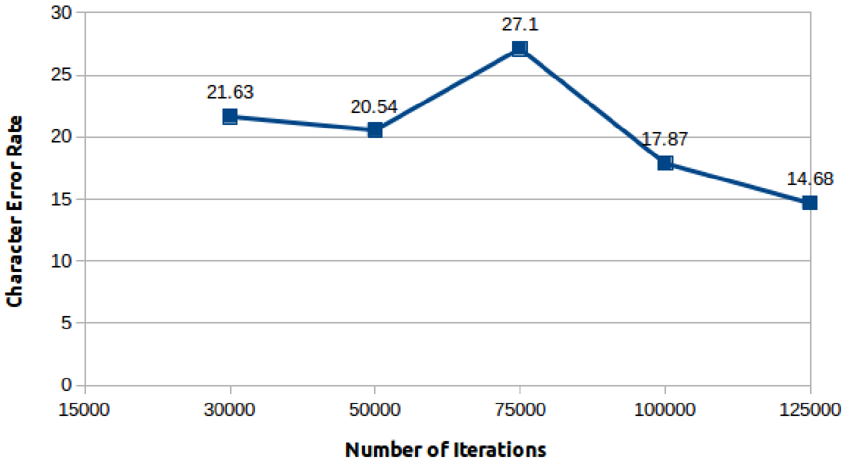


Figure 6.2: CER of LSTM in the first experiment for various training iterations.

In the first experiment, synthetic data of Appian's Roman history and the 687 images of the Greek Official Government Gazette have been used to train the LSTM-engine (on a total of 12,486 text-lines), while the text-lines of the Greek Parliament Proceedings are used as the test set (3,303 text-line images). It is important to note that in this setting the fonts in the training set are different from the four fonts of the test set. The LSTM recognizer yields a CER of 14.68%, after 125,000 iterations in the training phase, as detailed in Figure 6.2.

In the second configuration, the training set includes the synthetic data of Appian's Roman history, the text-line images of the Greek Official Government Gazette, and the text-line images of the three subsets of the Greek Parliament Proceedings (a total of 15,167 text-lines), while the text-lines of one subset of each of the above-mentioned sources are the test images (522 text-lines). In this experiment, the training data contain text written in five different fonts, while the test set includes one font that is unseen during the training. A CER of 5.67% is observed on the test data (see Figure 6.3).

In the last experiment, LSTM line recognizer is compared with the Tesseract and ABBYY OCR systems. For Tesseract, the training model for Greek polytonic script proposed by White [Whi13] is used. Regarding the ABBYY FineReader engine, we adapted it to the recognition of Greek polytonic scripts by adopting the procedure described in [SUHP$^+$15].

The 367 text-lines from Greek Official Government Gazette and the datasets of Appian's Roman history comprise the training data for the LSTM-based recognizer, while the remaining 2,836 text-lines of Greek Parliament Proceedings were included in the test data. The results are presented in Figure 6.4. It is worth mentioning that

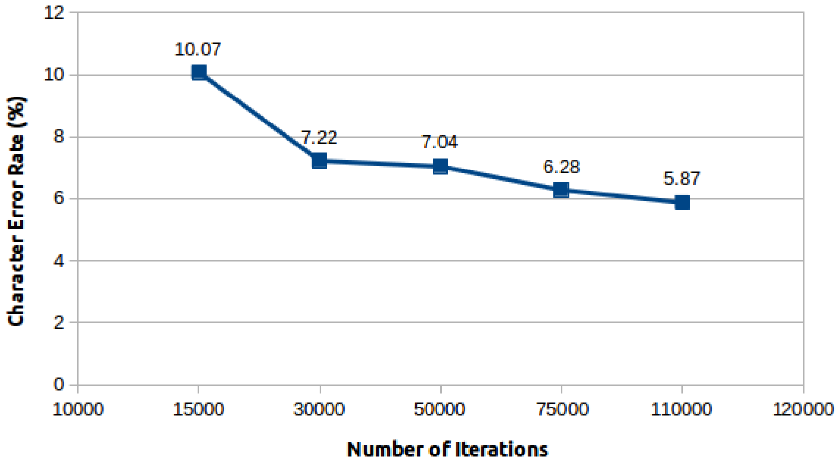


Figure 6.3: CER of LSTM in the second experiment for various iterations.

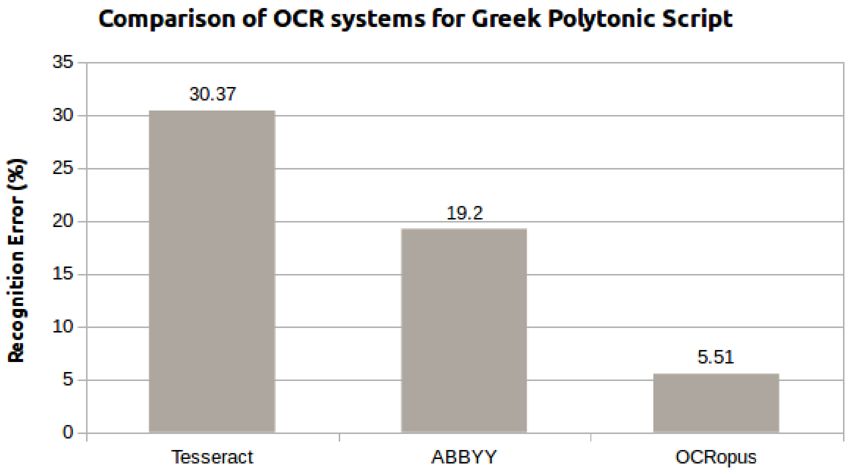


Figure 6.4: Comparison of three OCR systems for Polytonic Greek.

the poor performance of Tesseract is mainly explained by the fact that the characters' degradation in the test set is too high, and the character segmentation introduces too many mistakes that are propagated in the recognition stage. The results of Tesseract further strengthen the observation that many modern-day OCR system do not generalize well on unseen data, which may be totally different to that was used during the training.

Regarding the LSTM-based recognizer, the training model with the lowest error rate (0.16%) is produced after 138,000th iterations. This training model gives a CER of 6.04%. The total number of characters in the test set is 169,568. By using the training model produced after 148,000th iterations, with corresponding CER of 0.35%, results in reducing the CER on the test set from 6.05% to 5.51%.

The most frequent errors for the LSTM-recognizer are illustrated in Table 6.1. In

Table 6.1: Most frequent errors of the LSTM-recognizer

Number of errors	OCR result	GT character
109	–	΄
104	.	,
100	ι	τ
100	ο	σ
94	Ἐ	Ἑ
88	ἔ	ἕ
86	ι	_
79	ἐ	ἑ

particular, there are 318 deletion errors and 273 insertion errors out of 9,351 errors in total. Furthermore, there is a great number of errors where a letter is misclassified with the same letter but different accent. For example 94 occurrences of the letter *Ἐ* are erroneously classified as the letter *Ἑ*.

6.1.5 Conclusion

The results obtained on both Fraktur and ancient Greek demonstrate that the LSTM-based approaches generalize much better to unseen data than the previous ML approaches. In addition, during LSTM training, very low error rates on test data are observed, often long before one epoch of training has been completed, meaning that there has likely been no opportunity to "overtrain".

These results also show that training from artificially generated data is highly successful. One does, however, need to take care of generating this artificial data to resemble well with the scanned data, or else, the performance will suffer. However, to synthetically generate the training data in a particular script, one requirement is to possess some text in that script. This requirement, however, can not be fulfilled for ancient scripts where no such data is available. The next section reports the contribution of this thesis in dealing with the scarcity of GT training data for historical documents.

6.2 OCRoRACT: A Sequence Learning OCR System Trained on Isolated Characters

As mentioned previously that the contemporary ML approaches require a lot of transcribed training data in order to obtain satisfactory results. Transcribing the documents manually is a laborious and costly task, requiring many human hours and language specific expertise. This section presents a generic iterative training framework, named as ***OCRoRACT***[3], to address this issue. The proposed framework is not only applicable to historical documents but also suitable for present-day documents, where manually transcribed training data is unavailable. Starting with the minimal information available, the proposed approach iteratively corrects the training and generalization errors.

Specifically, a segmentation-based OCR method has been trained on individual symbols and used to OCR a subset of documents. These semi-corrected text-lines are then used as the GT data to train a segmentation-free OCR system, which learns to correct the errors by incorporating contextual information. The proposed framework is tested on a collection of 15^{th} century Latin documents with promising success. The iterative procedure using segmentation-free OCR is able to reduce the initial character error of about 23% (obtained from segmentation-based OCR) to less than 7% in few iterations.

6.2.1 Introduction

There could be multiple approaches to OCR documents, which can be either historical or modern, for which the training data is not available. The first intuitive idea to deal with the unavailability of data is to use a segmentation-based OCR approach that relies on character segmentation from a scanned image, and does not require much training data. The only requirement in such approaches is to extract unique characters from the whole document and train a shape-based classifier [AHN$^+$14, Whi13]. The next possible approach would be to manually transcribe a part of data that needs to be recognized, to train a classifier based on segmentation-free OCR approach (that are better for context-aware recognition), and then use this model to OCR the rest of the corpus.

However, both of the above-mentioned solutions carry their own demerits. The first approach, in practice, does not generalize well for new documents as shown by the performance of Tesseract default model for Polytonic Greek on Polyton-DB in Section 6.1.4. The second approach requires a large amount of transcribed data for every type of document that is to be digitized. The use of artificial data is also getting popular; however, generating such a database requires some already transcribed data.

---

[3] *OCRo* refers to **OCRo**pus and *RACT* refers to Tessa**RACT**

Our hypothesis is that the segmentation-based approaches can be used in tandem with the segmentation-free approaches to OCR documents where no or very limited training data is available. Specifically, the proposed approach is designed for 15^{th} century printed Latin documents, for which very small amount of training data is available. However, the proposed framework can be equally applied to other situations, where no training data is available.

The proposed framework successfully combines the benefits of both approaches. The first issue in using a *segmentation-free* approach is to have a manually transcribed data. This problem is solved by using a segmentation-based approach to generate semi-corrected GT data. The second issue in using a *segmentation-based* approach is the poor generalization on new data. LSTM networks aptly solve this problem, as they have demonstrated excellent results on unseen data [BUHAAS13].

The recognition error is reduced with each iteration of training using the proposed design. The CER is reduced to 6.569% from 23.64% in just three iterations. Section 6.2.2 describes the details of this framework.

6.2.2 Methodology

The novelty of the proposed idea is to use both segmentation-based and segmentation-free OCR approaches in tandem to design a high performance OCR system for documents having no or very limited GT data. The complete pipeline of OCRoRACT framework is shown in Figure 6.5.

The idea is to use the segmentation-based OCR to start the training on individual symbols, as not much training data is usually required for such systems. The OCR models obtained can be used to get a semi-corrected text, which can be used subsequently to train a segmentation-free OCR system.

The process starts with the extraction of individual symbols from scanned pages or text-lines. A language expert can provide a list of unique symbols in a given document along with their Unicode representation. Alternatively, a clustering process can follow the symbol extraction step to find the unique symbols. In the present work, the former path is taken due to the availability of some language experts. Tesseract [Smi07] is trained to recognize text, based on the given symbols. Tesseract OCR

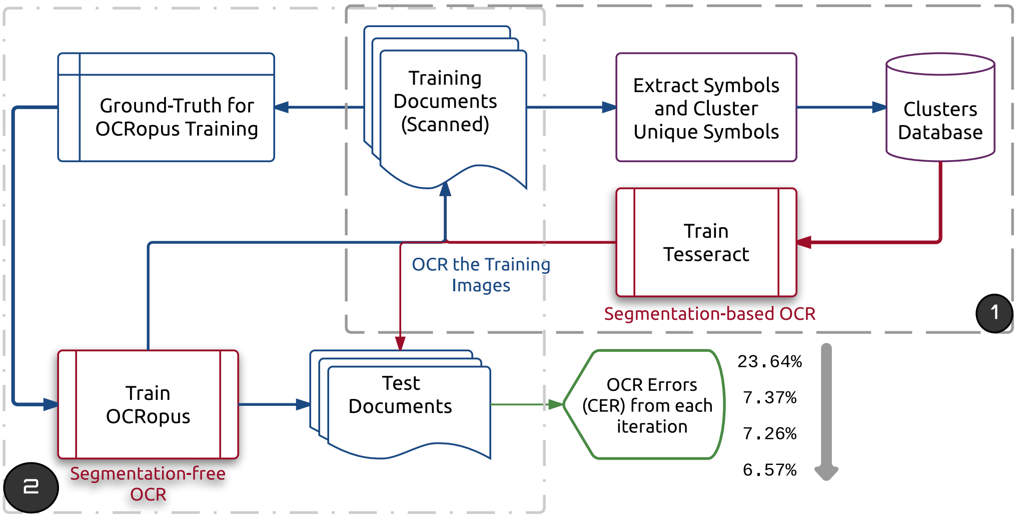

Ground-Truth for OCRopus Training
Training Documents (Scanned)
Extract Symbols and Cluster Unique Symbols
Clusters Database
Train Tesseract
OCR the Training Images
Segmentation-based OCR
1
Train OCRopus
Segmentation-free OCR
2
Test Documents
OCR Errors (CER) from each iteration
23.64%
7.37%
7.26%
6.57%

Figure 6.5: The process diagram of **OCRoRACT** system. In the first phase of process (shown in dashed box marked with ‘1’), unique symbols are extracted from the training corpus of scanned pages. These symbols are then used to train the Tesseract OCR system. The trained models obtained from the Tesseract are then used to generate semi-corrected GT information for the training of OCRopus OCR system. In the second phase (shown in dashed-dotted box marked with ‘2’), the OCRopus is iteratively used to train LSTM-based OCR models. The best model is susequently applied first to the test data and then to improve the GT data, which is used in the next iteration.

model is then used to generate the semi-corrected GT information, which is subsequently used to train OCRopus [OCR15], a segmentation-free open-source OCR system based on LSTM neural networks. The trained LSTM model is then used again to improve the GT information. This iterative process can be repeated for any number of iterations; however, in the current work, we chose to stop the training after seeing a small improvement (less than 1%) in the GT data from the previous iteration.

The details of LSTM-based OCR approach are presented in Chapter 5, and the details about the Tesseract can be found in [Smi07]. However, it is important to mention the procedure for training the Tesseract, which requires a specific *box-file* containing the bounding-box information of individual connected components in the image along with their corresponding Unicode representation. It is also required to provide this information in the form of a page, so that the Tesseract can learn the context during the training. As it has been mentioned before, that transcribed data for the medieval documents is not available; therefore, a meaningless text is generated from the unique symbols, that is, characters are placed in a text-line randomly. However, the characters are placed according to the placement rules with respect to baseline and x-height information. A thorough analysis has been carried out to find the statistics about various characters and the blank spaces between the individual characters and words.

The basic concept behind the proposed framework is to jointly use segmentation-based and segmentation-free approaches to OCR documents for which GT data is unavailable to train ML algorithms. There are three ways in which the proposed methodology has been evaluated. This is done to compare the results of various alternatives as mentioned in Section 6.2.1. Firstly, the Tesseract is used to OCR the database described in Section 7.5.2. Secondly, an LSTM network is trained directly on a subset of dataset that has been transcribed manually. Thirdly, the proposed approach is used to iteratively improve the semi-corrected GT data to train LSTM networks.

6.2.3 The Database

In order to validate our research hypothesis, 15th century Latin documents, available under the German government funded project, Kallimachos[4], are used. There are hundreds of novels written in Latin and Fraktur that are to be digitized. Firstly, it is also important to remember that no GT data is available for any of these novels. Secondly, the pages in these novels not only contain degradations because of aging but also contain annotations, which make them more challenging for document analysis.

In order to validate our research hypothesis, we have selected a subset of 100 images from one novel of Latin script for training, as well as 8 images from another Latin novel for testing. For performance evaluation, GT for 100 scanned pages with 3329 text-lines have been manually generated.

6.2.4 The System Parameters

The system parameters for Tesseract are kept to their default values. There are some tunable parameters to train LSTM-based OCR models. The first important parameter is the number of hidden layers, which is chosen to be one. The second important parameter is the number of LSTM cells at the hidden layer. Higher number of cells take longer to converge due to the enormous size of the network. For the experimental results reported in this section, the number of LSTM at the hidden layer is fixed to 100. Normalizing text-line images to a fixed height is an important preprocessing step. In our experience, the image height of 48 pixels is a reasonable choice and has worked satisfactorily for a variety of scripts. The learning rate and momentum are the remaining parameters and they are set to 1e-4 and 0.9 in the current work. The performance is estimated in terms of *Character Error Rate (CER)* which is defined by Equation 5.1.

---

[4] http://www.kallimachos.de/project/doku.php

6.2.5 Results

This section reports the results of the three experiments that are performed to test the performance of the proposed framework. There are two test datasets used in the evaluation process. The first dataset consists of two randomly selected pages from the same book that is used for training the Tesseract and OCRopus OCR systems; however, these pages are not part of any training. The total number of text-lines are 104 in these two pages and the total number of characters are 2877. This dataset is termed as 'T1'. The second dataset consists of 8 pages randomly selected from a book that has not been used in training at all. There are a total of 270 text-lines in these pages and the total number of characters are 7203 This dataset is termed as 'T2'.

The intermediate results obtained by the OCRoRACT framework are presented first before comparing the final results with other two alternatives. During the first stage (*iteration-0*), a Tesseract OCR model is trained on the unique symbols extracted from the given training documents. Tesseract yields a CER of 23.64% on 'T1' dataset and a CER of 14.4% on the data collection that is later used as the training database for LSTM-based OCR. This means that the LSTM-based OCR has been trained with a 14.4% erroneous GT data (*iteration-1*). The LSTM model thus trained outputs a CER of 7.37% on 'T1' database. The same model improves the GT data from having an error of 14.4% to 7.154% (an improvement of 50.1%).

During the second iteration (*iteration-2*), the LSTM models are trained with the improved GT obtained from the first iteration. The model thus trained gives a CER of 7.26% on the 'T1' dataset, and improves the GT by further 9.56% (CER of 6.47%). The improved GT is used again (*iteration-3*) to train another LSTM model in third iteration, which results in a CER of 6.57% on 'T1' data and improves the GT by 0.9%. This iterative process could go on further, but we decided to stop it at this stage as the GT improvement is now less than 1%.

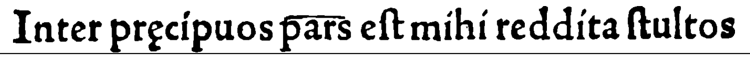
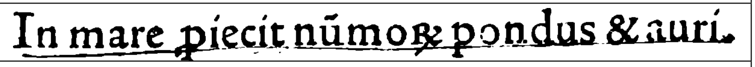
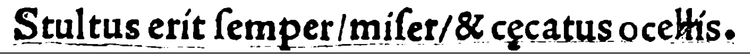
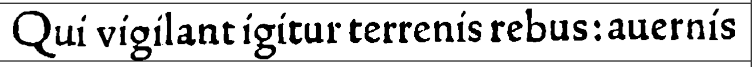
The LSTM models obtained after the third iteration are used to OCR the 'T2' dataset along with Tesseract (that was trained at the beginning of this process) and the OCRopus model, trained with correct GT information. The results of the three evaluations are listed in Table 6.2.

A qualitative comparison is also performed and some of the input images along with the OCR output are shown in Table 6.3.

Table 6.2: Comparison of applying Tesseract, OCRopus and OCRoRACT. The results obtained by the OCRopus seems the best, but it must be noted that this system was trained with the correct GT information, while the OCRoRACT is trained with semi-correct GT information. Noteworthy is the results obtained on 'T2' dataset where the difference between the OCRopus and OCRoRACT is relatively small.

	Character Error Rate, CER (%)		
Dataset	**Tesseract**	**OCRopus**	**OCRoRACT**
T1	23.64	1.8	6.57
T2	25.21	10	10.6

Table 6.3: Qualitative analysis of three OCR system on 'T2' test data. Tesseract does not compete well with other two OCR systems. Both OCRopus and OCRoRACT perform similar to each other; however, OCRopus results are a little better than OCRoRACT. It must be noted, however, that OCRoRACT has been trained with erroneous GT and the OCRopus with the correct GT.

Image	
Tesseract	Inter pręcīpuos p̄aTs eſt mihi reddita ſtultos
OCRopus	Inter pręcipuos par eſt mihi reddita ſtultos
OCRoRACT	Inter pręcipuos ars eſt mihi reddita ſtultos
Image	
Tesseract	In mare ꝓieęcq9Lꝗūmꝗꝑęꝓęoꝗdus&fluti.
OCRopus	In macōꝓiect nūmoʀ| pondus&auri.
OCRoRACT	qatꝓiecit nūmo&pondus &auti.
Image	
Tesseract	Stultus ęcq̃ę ſeqꝫꝑner/mVāſct/&ęcęcuatusocekffis.
OCRopus	Stultus erit ſemper/miſer/& cęcatusoceffis.
OCRoRACT	Stultus erit ſemper/miſer/& cęcatus ocefis.
Image	
Tesseract	Qui vigilant igicur terrenis rebus:auernis
OCRopus	Qui vigilant igitur terrenis rebus:auernis
OCRoRACT	Qui vigilant igitur terrenis rebus:auernis

6.2.6 Error Analysis and Discussions

The top confusions in applying the three OCR systems on 'T1' dataset are shown in Table 6.4. Tesseract has mostly made errors in recognizing between small and capital

Table 6.4: Top errors generated by the Tesseract, OCRoRACT and OCRopus OCR systems when applied on ‘T1’ dataset. **GT** denotes the ground-truth, **Pred** denoted the predicted output and **Error** shows the number of errors. _ shows the *insertion* or *deletion* of the corresponding character. Total number of errors made by a particular OCR system are shown below the name of that system.

Tesseract			OCRoRACT			OCRopus		
Total Errors		680	Total Erros		189	Total Errors		53
GT	**Pred.**	**Errors**	**GT**	**Pred.**	**Errors**	**GT**	**Pred.**	**Errors**
	_	18	_	i	19	_		7
_		14	_		17		_	3
1		11		_	13	_	l	3
S	s	8	$\bar{a}$	a	7	_		2
$\bar{r}$	i_	7	$\bar{u}$	u	5	$\bar{u}$	u	2
f	i	6	l	ſ	4	.	_	1
I	l	5	.	_	4	ſ	_	1
t	r	5	i	_	4	l	ſ	1

letters. This is understandable as these errors normally occur when contextual information is not present. It should also be noted that the usual application of Tesseract requires an associated language model. The same has not been done in the current work as no language model for medieval Latin is available.

The OCRoRACT system, which has been trained on the erroneous GT data corrects many of these errors. However, it confuses between similar shape characters like ‘$\bar{a}$’ and ‘a’ or ‘$\bar{u}$’ and ‘u’. It is to be noted that both of these systems yield many *insertion* and *deletion* errors related to the ‘space’ character. One way to reduce the errors made by the OCRoRACT system is to retrain Tesseract system with more variety of symbols that are causing confusions.

The OCRopus system produces less errors; however, the top confusions are again ‘space’ *insertion* and *deletion*. Other notable errors are deletion errors. The *deletion* errors can generally be reduced by better training. It must be noted that the training data is not much and the performance of LSTM networks in still quite satisfactory.

It is also important to see the errors obtained from applying these three systems on ‘T2’ dataset, which has not been used during the training. The top confusions are listed in Table 6.5. By comparing the outputs of three systems on ‘T2’ dataset from

Table 6.5: Top errors generated by the Tesseract, OCRoRACT and OCRopus OCR systems when applied on 'T2' dataset. **GT** denotes the ground-truth, **Pred** denoted the predicted output and **Error** shows the number of errors. _ shows the *insertion* or *deletion* of the corresponding character. Total number of errors made by a particular OCR system are shown below the name of that system.

Tesseract			OCRoRACT			OCRopus		
Total Errors		1769	Total Erros		751	Total Errors		704
Pred.	**GT**	**Errors**	**Pred.**	**GT**	**Errors**	**Pred.**	**GT**	**Errors**
	_	72	_		72	_	ſ	142
_		48	_	i	38	_		29
t	r	25		_	36	_	l	20
u	n	23	t	r	16	.	_	16
l	_	21	.	_	13		_	13
.	_	14	$\bar{a}$	$\bar{e}$	10	.	:	7

Table 6.5, it can seen that Tesseract and OCRoRACT make similar "shape confusion" errors. However, the top errors for OCRopus are due to the misrecognition of characters. This shows that it is difficult for OCRopus to generalize well to unseen variety of characters that it has not seen during the training.

It is also evident from the results that the OCRoRACT performs very well on unseen data. The iterative nature of this framework allows for the addition of further training data (albeit erroneous) into the system, thereby, resulting in better training.

One drawback of this system is its use for cursive scripts like Arabic and Devanagari, where character/ligature segmentation is non-trivial. Arabic-like scripts contain a huge amount of ligatures and the reliable segmentation of ligatures requires more research efforts. However, one possible solution for such scripts is to use the artificial data instead of using the segmentation-based approach to start the training. The LSTM models thus obtained could be used to OCR the scanned documents.

6.2.7 Conclusion

This section presents a novel framework, the *OCRoRACT*, to combine the benefits of both segmentation-based and segmentation-free OCR approaches to OCR documents where no training data is available. Moreover, the reliance on a human expert to generate the transcribed data is greatly reduced, if not eliminated by using the proposed methodology. The performance of this system is excellent when a document with similar script is evaluated. Moreover, since the system does not require accurate GT information, it can be retrained as new output becomes available. This framework can be improved further by incorporating the ability of the system to be used for cursive scripts having huge amounts of ligatures.

6.3 Chapter Summary

This chapter discusses the contribution of this thesis to OCR of historical documents. LSTM-based OCR methodology has successfully been applied to three historical European scripts, Fraktur, Polytonic Greek and Medieval Latin of 15th century. Creating a manual GT data for LSTM-based approaches is tedious and costly in terms of manual efforts and language expertise. The use of synthetic data for LSTM training has been very successful to achieve reasonable accuracy on Fraktur and Greek; however, to generate the artificial data, one must have already transcribed text. To overcome the challenge of unavailability of GT data, a novel framework, *OCRoRACT*, has been introduced combining the segmentation-based and segmentation-free OCR approaches. Segmentation-based approaches require individual characters for training; however, they do not generalize well in practice on new data. This limitation is overcome in the proposed framework by training the LSTM networks on the semi-corrected GT data generated by the segmentation-based approach.

Chapter 7

Sequence Learning for Multiple Script Identification

Script identification is a crucial step in digitizing multilingual documents. In addition to simplifying the OCR process, it could be used for many other tasks like document indexing, routing, and categorization of documents [PD14]. The traditional process to OCR documents with multiple scripts is to identify and separate the scripts before applying the OCR models. By doing so, the problem is reduced to applying single-script OCR models to the separated scripts.

A standard script identification approach is to manually extract salient features of the given script(s) and then apply a pattern classification algorithm. However, contemporary ML techniques can learn the distinguishing features automatically; thereby eliminating the manual feature extraction step. Sequence learning is based on the sequence-to-sequence matching principle where a neural network is trained such that the input-output pair is comprised of sequences (text-lines in the case of documents).

The contribution of this thesis[1] in using sequence learning for multiple script identification is described in this chapter. The highlights of the proposed approach are:

- No script-specific handcrafted features are extracted. The LSTM networks have yielded excellent results using only the raw pixel values.
- No word- or character-segmentation is required (text-lines extraction is however essential); yet the script identification is done at character level.
- The LSTM networks are used for sequence-to-sequence mapping.

---

[1] This work is based on author's work report in [UHAS$^+$15]

This chapter is further organized as follows: Section 7.1 describes the scope of this chapter with respect to the categorization of multilingual documents given in Chapter 3. Sections 7.2 and 7.3 report various script identification techniques used by other researchers. Section 7.4 details the LSTM-based sequence-learning technique for multiple script identification. Section 7.5 describes the experimental setup, database used, and performance metrics; while Section 7.6 shows the results of experimental evaluation and discusses the errors in detail. Section 7.7 concludes the chapter with a short summary suggesting few possible future directions in which this work can be extended.

7.1 Scope

A general categorization of multilingual documents has been described in Chapter 3. The first class of multilingual documents are those which have different languages sharing the same script, while the second type are those in which a single language is written in two or more scripts. The documents in the third category contain different languages and different scripts, for instance, English-Arabic, English-Greek, and Chinese-Urdu. For such documents, visual appearance of underlying scripts is highly discriminative and multiple scripts can be separated more easily as compared to the documents in other two categories.

This chapter demonstrates the capacity of LSTM networks in simplifying the script identification task by 1) removing the feature extraction step, and 2) assigning script labels at character level despite processing the whole text-line.

7.2 Heuristic-Based Approaches for Script Identification

Pioneering work on multiple script identification was done by Spitz [Spi97]. They used optical density distribution as a feature for the Han script and used typographic character codes for Latin-based script identification. In an another early work, Pal and Chaudhuri [PC02] developed a script identification system to recognize printed Roman, Chinese, Arabic, Devanagari, and Bangla text lines. They used different features to identify a particular script.

Ghosh et al., in [GDS10], reported a comprehensive review on multiple script recognition. They divided the script recognition process into two categories of structure- and appearance-based approaches and provided comparative analysis of numerous systems. Appearance-based approaches are global in the sense that they extract text-blocks irrespective of the document's script. These techniques are usually faster and are applicable to a wide variety of scripts. Structure-based are local approaches, which are useful where script identification is required at line-, word- or character-level. Both approaches require segmentation of text blocks at the preprocessing stage.

Wang et al. [WLS10] proposed a noise tolerant script identification system based on pixel distribution of low-resolution skeleton of a character or a sub-word. This feature discriminated Chinese, Korean, Japanese, and English scripts at connected component level. Padma and Vijaya [PV10] proposed a texture-based global approach based on Wavelet packet feature (Haar basis function). They reported the script identification accuracies for seven scripts used in India.

Sharma et al. [SCPB13] proposed a methodology for script identification in video frames. They used three well-known features, namely, Zernike moments, Gabor, and gradient features. A script was recognized using Support Vector Machines (SVM) classifier on these features. They evaluated their method on English, Hindi, and Bangla scripts. In order to overcome the problems associated with video frames, they used super-resolution and skeletonization as preprocessing steps.

Rani et al. [RDL13] proposed a script identification method for pre-segmented English and Gurmukhi characters. They extracted Gabor features and gradient features from the characters and applied SVM classifier for recognition.

Recently, Echi et al. [ESB14] proposed a script identification technique for both printed and handwritten text for Arabic-Latin documents. They used many known features in addition to their own features to identify Arabic and Latin. They compared the performance of several classifiers including Naive Bayes, k-NN, Decision Trees, Multilayer perceptron (MLP) and SVM on selected features.

All of the above-mentioned methods rely on extraction of sophisticated features and image processing techniques along with various heuristics. This is because of the common belief that script-related features are essential to be extracted for reliable script identification. Therefore, major effort has been spent by the research community to find and fine-tune script-related features. One major issue with all feature-based methodologies is the extensive use of heuristics to adapt these methods to the task at hand. Therefore, techniques designed for one script or a set of scripts are not generally suitable for other scripts.

7.3 Machine Learning Based Approaches

Contemporary Machine Learning (ML) techniques, like Deep Belief Networks (DBN) and deep Convolutional Neural Networks (CNN), are gaining popularity in the fields of Computer Vision (CV) and Pattern Recognition (PR), and one of the reason for their wide application in this domain is that they do not rely on sophisticated handcrafted features. Instead they learn the features from the given data themselves, and still perform better than the methods based on handcrafted features. Likewise, there has been some recent efforts in applying these techniques for script identification.

Rashid et al. [Ras10] proposed a script identification method based on CNN, which learns the shapes of individual connected components without explicitly extracting features from them. They tested their method on Latin/Greek, Latin/Arabic and Antiqua/Fraktur scripts. However, they applied a post-processing step to assign small diacritics to corresponding base shapes to improve the final accuracy.

Genzal et al. [GPTF13] proposed an HMM-based script identification system. On a private database consisting of 54 languages with 18 different scripts, they reported an average script identification error rate of 1.73%. The text-lines in their work were assigned a dominant script if it had more than one script.

Singh and Jawahar [SJ15] recently proposed an RNN-based approach for script and language separation. Their approach identifies words of different scripts and languages in a text-line. They cut the word image into upper and lower half for profile feature extraction. The result on a large corpus containing 15 languages shows an average accuracy of 93.96% at word-level and 97.9% at text-line level.

Although all of the above-mentioned techniques are based on modern machine learning principles, but the first work applies post-processing step to improve the accuracy, the second approach achieves low recognition error by the use of carefully selected features and the third work employs profile features of words for the recognition. Our approach, however, is very simple as we neither extract any feature(s) nor we have applied any kind of post-processing on the output. The details of this method are described in the following section.

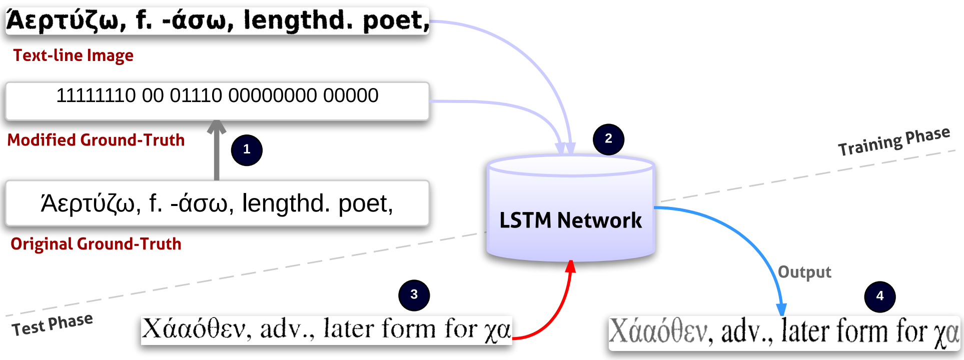


Figure 7.1: Complete pipeline to use LSTM networks for multiple script identification. (1) The Ground-Truth (GT) information of a given text-line in a training set is modified to a string of class labels. (2) The LSTM network learns individual scripts using the modified GT as target labels for the individual characters. (3) During the **test phase**, the test text-line image is segmented into individual scripts. Noteworthy, during the test phase, trained LSTM emits a sequence of class labels along with their approximate locations. (4) The location information then is used to segment the input text-line image into respective script-images. In the shown sample image, the gray color shows the Monotonic Greek script, while the black color shows the English (Latin) script.

7.4 LSTM Networks for Script Identification

The scheme introduced in this chapter models the script identification as a multi-class classification problem. A single target class-label is assigned to all characters of a particular script, which the LSTM network learns in a supervised classification paradigm. A text-line is not segmented into individual characters; instead LSTM network learns scripts' shapes in a sequence-to-sequence mapping manner. In other words, we are training the LSTM network to learn a particular target class-label for all alphabets in a given language/script.

Figure 7.1 shows the proposed script identification process. The open source OCRopus document analysis system [OCR15] is used for the experimental evaluation. The OCRopus line recognizer is basically designed for OCR task, where the goal is to transcribe a given text-line; therefore, the proposed idea essentially adapts this OCR system to perform the script recognition task. The modification made in the current design to adapt the OCR system is very simple. Instead of giving the Ground-Truth (GT) information about the individual characters in a text-line, a sequence of class-labels is provided as the GT at the training stage . To achieve this in practice, the GT information is modified such that it represents a two-class classification problem (see Figure 7.1). The modified GT information is used as the target sequence for a given text-line image during the training. LSTM network learns the class association of each characters based on their own shape as well as the contextual information of

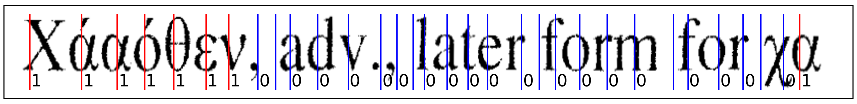


Figure 7.2: The location information predicted by the trained LSTM model on a test text-line image. The output from the LSTM trained model for script identification is shown in vertical lines with associated class labels. The vertical line is at the approximate location of the character predicted by the LSTM line-recognizer. This information is then used to separate the two scripts.

surrounding shapes. At the test stage, the trained network produces a script label corresponding to individual character in the text-line. It produces a string of labels corresponding to each character as shown in Figure 7.2. The network also predicts the approximate location of individual characters, which in turn, is used to separate the scripts.

The proposed approach for multiple script identification is designed to work on text-line images directly; it is therefore not necessary to extract individual characters or words from the page or text-line images, as done in most of the other reported approaches. Only text-line extraction is required from a full given page.

Moreover, unlike the usual approaches, the raw pixel values have been used as the only features. This is done because the LSTM networks, like other contemporary machine learning techniques, have shown to differentiate between many classes based on just raw pixel values [BUHAAS13].

7.5 Experiments

7.5.1 LSTM Networks' Architecture

For the experiments reported in this chapter, 1D-LSTM architecture has been employed. It is the same as described in [BUHAAS13]. In 1D-LSTM networks, a sliding window of 1-pixel width and of height equal to the image height, traverses the input text-line image to convert it into a one dimensional sequence. The height of the image is termed as "depth" of the 1D sequence and each slice of the image thus obtained is called a "frame". It is necessary to make the depth of all images equal, so that the individual frames in each image are equal. This process of making heights equal is labelled as "normalization", and it is essential for such types of networks. For this work, we use the same normalization method used in [YSBS15] and it is available in the OCRopus system. This method of normalization, which can be used for many other scripts, is based on filter operations and affine transformations.

The LSTM network consists of one input layer, one hidden layer containing 50 LSTM cells and one output layer. For 1D-LSTM training, a sliding window of 48 × 1 scans the input image and each frame is given to the LSTM network for training. The network learns to classify each frame into one of the target classes, including the reject class, based on the contextual information of that frame. The bidirectional mode of LSTM architecture is used to access the context in both forward and backward direction. In bidirectional mode, two hidden layers scan the input. One traverses the input from right-to-left and the other from left-to-right and both layers are connected to a single output layer.

7.5.2 Database

There is no standard database available for script identification purpose. Therefore, a synthetic database consisting of text-line images from freely available English-Greek text has been developed to evaluate the proposed method. A text-line generation tool available in OCRopus has been used for this purpose. This tool can generate any number of text-line images with a given text-file and font-files. A modification in this utility is done to generate the GT data for each transcription that suits our needs. 90,000 text-line images are synthetically generated in a variety of fonts (both serif and sans-serif) using degradation models described in Section 4.3. A separate test database consisting of 9,500 text-line images from a different English-Greek text has also been prepared. This database is available free of cost for the research purposes from the author.

7.5.3 Performance Metric

Since we have adapted an OCR system for the script identification purpose, the output is like a transcription. Therefore, the performance is measured in terms of edit distance (or *Levenshtein Distance*). Section 5.1.3 describes the details on this metric.

7.6 Results and Discussions

There are a total of 9,500 text-line images in our test dataset consisting of 289,013 characters. LSTM network trained for script-identification task as described in Section 7.4 yielded an accuracy of 98.19% (an error rate of 1.81%). Some sample test images are shown in Figure 7.3. Top 15 confusions are shown in Table 7.1. It should

Table 7.1: Top 15 confusions for script identification task.

GT	**No.**	**GT**	**No.**	**GT**	**No.**
v	749	o	717	ε	552
ι	287	ρ	277	a	228
u	210	τ	174	x	149
φ	136	λ	135	η	128
ω	120	σ	120	γ	16

GT – Ground-truth,
No. – No. of occurrences

be noted that errors made by punctuation marks are not considered as they are generally script independent.

From Table 7.1, one can see that many top confusions are due to the similarity in shapes of characters between the two scripts, like ‘v’ with ν, ‘o’ with ο, ‘i’ with ι, ‘p’ with ρ, ‘a’ with α, ‘u’ with υ, ‘t’ with τ and so on. These type of errors will not be present when characters of both scripts are significantly different, e.g., English and Hindi or Arabic and Chinese. A bigram or trigram analysis could be done to further investigate the causes of such confusions.

Many confusions occur at the locations where the script of the document changes from English to Greek or vice versa. For example, in Figure 7.2, the second last character χ is misclassified as ‘X’ because the preceding character is an ‘r’ (English). Contextual information at that time-step incorrectly makes the LSTM network classify this character as belonging to the Latin script. Again, such errors will not occur in documents in which character shapes are considerably different.

Script identification has been reported by many researchers, but the unavailability of a standard database hampers the comparison of true performance among reported methodologies. Moreover, the difference in performance metrics also obstructs a fair comparison.

The results also show the powerful learning capabilities of LSTM networks. LSTM networks are able to learn many variation in shapes for the same class, for example, it could recognize the class association of whole English and Greek scripts to a single class-label. This suggests us that these networks are very useful for large-scale classification problems where class members show large variations.

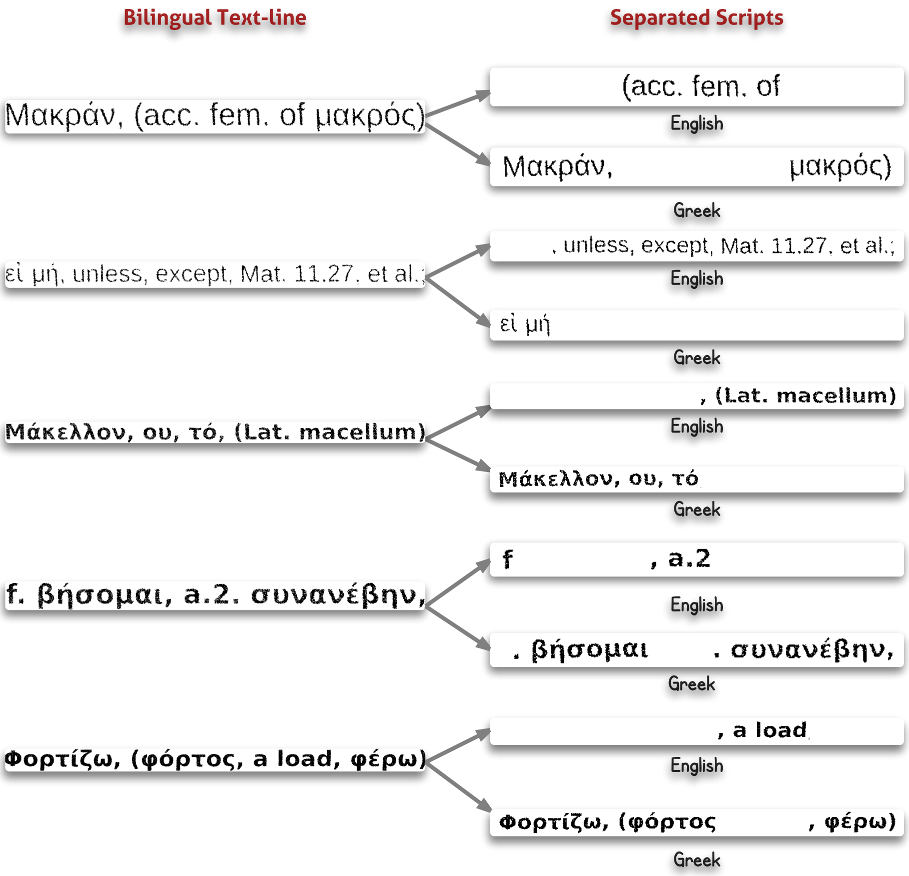

Bilingual Text-line
Separated Scripts
Μακράν, (acc. fem. of μακρός)
(acc. fem. of
English
Μακράν, μακρός)
Greek
εἰ μή, unless, except, Mat. 11.27, et al.;
, unless, except, Mat. 11.27, et al.;
English
εἰ μή
Greek
Μάκελλον, ου, τό, (Lat. macellum)
, (Lat. macellum)
English
Μάκελλον, ου, τό
Greek
f. βήσομαι, a.2. συνανέβην,
f , a.2
English
. βήσομαι . συνανέβην,
Greek
Φορτίζω, (φόρτος, a load, φέρω)
, a load
English
Φορτίζω, (φόρτος , φέρω)
Greek

Figure 7.3: Samples of some text-lines separated by LSTM-based script identification method. It is important to note that errors resulted from the punctuation marks including parentheses are not considered as these marks are generally script independent.

7.7 Chapter Summary

This chapter introduces a new method for identifying multiple scripts in multilingual documents. LSTM-based line recognizer has been trained to learn class associations of individual characters for a given bilingual text-line. Experimental evaluation on bilingual English-Greek data has shown encouraging results. This work can be extended further in numerous ways. Firstly, other bilingual documents such as English-Devanagari, Arabic-Latin, or English-Chinese can directly benefit from this approach. Secondly, this approach can be adapted to documents with more than two scripts. Another direction in which the research can be extended is that the LSTM networks can be used to directly OCR the multilingual documents. They have demonstrated a good capacity to learn a lot of classes at the same time; this means that we can avoid the script identification step in some, if not all, multilingual documents.

Chapter 8

Generalized OCR Framework for Multilingual Documents

Multilingual documents are common in the computer age of today. Plethora of these documents exist in the form of translations, books, operational manuals, etc. The abundance of these multilingual documents in everyday life is observed today due to two main reasons.

Firstly, technological advancements are reaching in each and every corner of the world due to globalization, and there is an increasing need from the international customers to access the technology in their native language. This phenomenon has a two-fold impact: 1) operational manuals of electronic gadgets are required to be in multiple languages, 2) the access to knowledge available in other languages has become very easy; thereby, an increase in bilingual books and dictionaries has been witnessed.

Secondly, English has become an international language, and the effect of this internationalization is evident by its impact on many languages. Several languages have adopted words from English and various documents, for instance newspapers, magazines, and articles, use many English words on a daily basis. Therefore, the need to develop reliable Multilingual OCR (MOCR) systems to digitize these documents has inflated manifold.

Despite the increase in the availability of multilingual documents, automatic recognition of multilingual text remains a challenge. Popat [Pop12] pointed out several challenges in the context of the Google books project[1]. Some of these unique challenges are:

---

[1] http://en.wikipedia.org/wiki/Google_Books

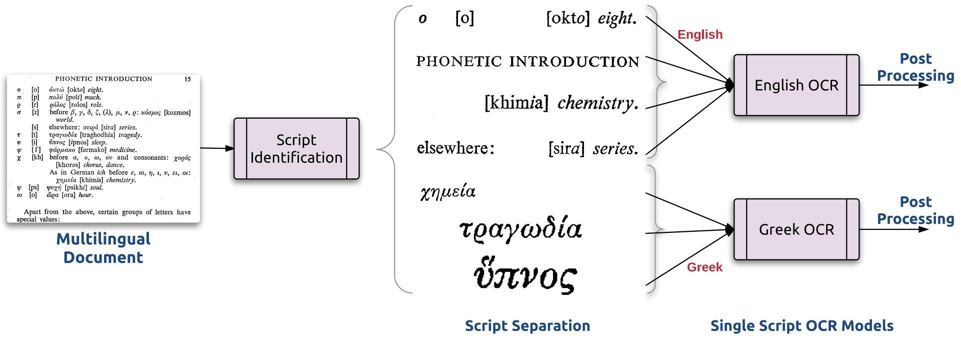

PHONETIC INTRODUCTION
Multilingual Document
Script Identification
o [o] [okto] eight.
PHONETIC INTRODUCTION
[khimia] chemistry.
elsewhere: [sira] series.
χημεία
τραγωδία
ὕπνος
English
Greek
English OCR
Greek OCR
Post Processing
Post Processing
Script Separation
Single Script OCR Models

Figure 8.1: The traditional approach used to OCR multilingual documents. A separate script identification step is employed to identify and separate the multilingual text. After separating the scripts, a single script OCR model is applied to finally recognize the text.

- Multiple scripts/languages on a single page.
- Multiple languages in same or similar scripts, like Arabic-Persian, English-German.
- The same language in multiple scripts, like Urdu in Nastaleeq and Naskh scripts.
- Archaic and reformed orthographies, for example, 18^{th} Century English, Fraktur (historical German).

One solution to handle multilingual documents is to develop an OCR methodology that can recognize all characters of all scripts. However, it is commonly believed that such a generic OCR framework would be very difficult to realize [PD14]. The alternate process (as shown in Figure 8.1) is to employ a script identification step before recognizing the text. This step separates various scripts present in a document, so that a unilingual OCR model can be applied to recognize each script. This procedure, however, is unsatisfactory for many reasons, some of which are listed below:

- The script identification is itself quite a challenging feat. Traditionally, it involves finding suitable features of the given script(s). One has to either fine tune these hand-crafted features or has to look for some other features, if the same script identification methodology has to be used for other scripts.
- The process of script identification (see chapter 7) is not perfect, thereby the scripts recognized by such process can not be separated reliably. This directly affects the recognition accuracy of the OCR system employed.

- Moreover, humans do not process the multilingual documents using the script identification step. A person possessing multilingual prowess reads a multilingual document in a similar manner as he/she would read a monolingual document.

Hence the ultimate aim to OCR multilingual documents is to develop a generalized OCR system that can recognize all scripts. An MOCR system must be able to handle various scripts as well as it should be robust against the intraclass variations, that is, it should be able to recognize the letters despite slight variations in their shapes and sizes.

Although the idea of generalized OCR system is not new, it has not been pursued greatly because of lack of computational powers and suitable algorithms to recognize all characters of multiple scripts. However, recent advancement in machine learning and pattern recognition fields have shown great promise on many tasks that were once considered very difficult. Moreover, these learning strategies are claimed to mimic the neural networks employed in the human brain. So they should be able to replicate the human capabilities in a better way than other neural networks.

The main contribution of this chapter is a **Generalized OCR** framework[2] that can be used to OCR multilingual and multiscript documents such that there is no need to employ the traditional script identification step. A sub-goal of this work is to highlight the discriminating power and sequence learning capability of LSTM networks for a large number of classes for OCR tasks. The trained LSTM networks can successfully discriminate hundreds of classes when it is trained for multiple scripts/languages simultaneously.

The rest of this chapter is organized as follows. Section 8.1 reports the work done by other researchers to develop generalized OCR systems for multilingual documents. Our quest for a generalized OCR system starts with the development of a single OCR model that can recognize multilingual text in which all languages belong to a single script. Section 8.2 discusses the cross-language performance of LSTM networks. The next step of our quest is to extend the idea of "single OCR model" from multilingual documents to multiscript documents. A single OCR model that can recognize text in multiple scripts is the first step in realizing a generalized OCR system. Section 8.3 describes the design of LSTM-based generalized OCR framework in detail. Section 8.4 concludes the chapter with a brief summary and outlines some directions in which the present work can be further extended.

---

[2] We followed the use of term "Generalized OCR" for the proposed methodology from [PD14].

8.1 Traditional Approaches for MOCR

The usual approach to address the MOCR problem is to somehow combine two or more separate classifiers [OHBA11]. This is because of the common belief that a reasonable OCR output for a single script can not be obtained without sophisticated post-processing steps such as language modeling, use of dictionary to correct OCR errors, font adaptation, etc. Natarajan et al. [NLS+01] proposed an HMM-based script-independent MOCR system. Feature extraction, training and recognition components of this system are all language independent; however, they used language specific word lexicon and language models for the recognition purpose.

There have been efforts reported for the adaptation of the existing OCR systems to other languages. Open source OCR system Tesseract [SAL09] is one such example. The recognition of characters in Tesseract is based on hierarchical shape classification. The character set is reduced to few basic characters and then at last stage, the test sample is matched against the representative of the reduced set. Although, Tesseract can be used for a variety of languages, it can not be used as an all-in-one solution in situations where multiple scripts are present in a single document together. Similar to the Tesseract OCR, BBN BYBLOS system [LBK+98] can be trained for multiple languages; however, this system is also not capable of recognizing multiple languages and scripts simultaneously.

To the best of our knowledge, not a single method has been proposed for MOCR, that can achieve very low error rates without using sophisticated post-processing techniques. However, experiments on many scripts using LSTM networks have demonstrated that significant OCR results can still be obtained without such techniques. The details about the LSTM-based language independent OCR framework are presented in the next section.

8.2 Language Independent OCR with LSTM Networks

Language models or recognition dictionaries are usually considered an essential step in OCR. However, using a language model complicates the training of OCR systems, and it also narrows the range of texts that an OCR system can be used with. However, recent results have shown that LSTM-based OCR yields low error rates even without language modeling. This leads us to explore the question as to what extent LSTM models can be used for MOCR without the use of language models. To this end, we measure[3] cross-language performance of LSTM models trained on different languages. They have exhibited a great capability to be used for language independent OCR. The recognition errors are very low (around 1%) without using any language model or dictionary correction.

Our hypothesis for language independent OCR is that if a single model can be obtained for a single script which is common to many languages, e.g. Latin, Arabic, Devanagari, we can then use this single model to recognize text of that particular family. Doing so, the efforts to combine multiple classifiers can be reduced.

The basic aim in this work is to benchmark how LSTM networks use language modeling to predict the correct labels or can they do better without using any language modeling and other post-processing step. Additionally, we also want to see how well LSTM networks use the contextual information to recognize a particular character.

8.2.1 Experiment Setup

To explore the cross-language performance of LSTM networks, a number of experiments have been performed. We have trained four separate LSTM networks for English, German, French and a Mixed-Data of all these languages. For testing, there are a total of 16 permutations. Each of the four aforementioned LSTM model is tested on the respective language and on the other three languages as well, for example, testing LSTM network trained on German language on a separate German corpus, French, English, and Mixed-Data. These results are detailed in Section 8.2.5.

As an error metric, the ratio of insertions, deletions and substitutions relative to the GT (CER) has been used and accuracy is measured at the character level. This error metric is termed as 'Levenshtein Distance' in the literature and is given by Equation 5.1.

This section is further organized as follows. The next sub-section describes the binarization and the text-line normalization, which are the first step in the LSTM-based approach. Details on preparing the dataset for training the LSTM models and the dataset for evaluation are given next. After the details on the database, LSTM network parameters are given, while the results are presented at the tail end of this section.

---

[3] This section is based on the author's work reported in [UHB13]

8.2.2 Preprocessing

Binarization and text-line normalization form the preprocessing step in this experiment. Since synthetically generated text-lines are used in this work, binarization is carried out at the text-line generation step. However, text-line normalization is done separately. Scale and relative position of a character are important features in distinguishing characters in the Latin script (and some other scripts). Moreover, 1D-LSTM networks are not translation invariant in vertical direction. The text line normalization is therefore, an essential step in applying such networks. In this work, we have used the normalization approach introduced in [BUHAAS13] (see Appendix B for details), namely text-line normalization based on a trainable, shape-based model. A token dictionary created from a collection of text lines contains information about x-height, baseline, and shape of individual characters. These models are then used to normalize any text-line image.

8.2.3 Database

To evaluate the proposed methodology, a separate synthetic database for each language is developed using the approach described in Chapter 4. Separate corpora of text-line images in German, English and French languages are generated with commonly used typefaces (including bold, italic, italic-bold variations) from freely available online literature. These images are degraded using some of the degradation models described in [Bai92] to reflect the scanning artifacts. Four degradation parameters namely *elastic elongation*, *jitter*, *sensitivity*, and *threshold* have been selected. Sample text-lines images from our database are shown in Figure 4.6. Each database is further divided into training and test subsets. Statistics on the number of text line images in training and test corpora of each script are given in Table 4.2.

8.2.4 LSTM Architecture and Parameters

For the experiments carried out in this work, 1D-BLSTM architecture has been utilized. We have found that this architecture performs better than more complex LSTM architectures for printed OCR tasks (please refer to Appendix-A for further details about the LSTM networks and its different variants). 1D-LSTM networks require text-line images of a fixed height as they are not translation invariant in the vertical dimension. Therefore “normalization” is employed to make sure that the sequence depth remains consistent for all the inputs.

Table 8.1: Experimental results of applying LSTM networks for MOCR. Note that the error rates of testing LSTM network trained for German on French and networks trained for English on French and German are obtained by ignoring the words containing special characters (umlauts and accented letters). LSTM networks trained for individual languages can also be used to recognize other scripts, but they show some language dependence. All these results are achieved without the aid of any language model.

Model / Script	English (%)	German (%)	French (%)	Mixed (%)
English	**0.5**	1.22	4.1	1.06
German	2.04	**0.85**	4.7	1.2
French	1.8	1.4	1.1	**1.05**
Mixed-Data	1.7	1.1	2.9	**1.1**

The text lines are normalized to a height of 32 in the preprocessing step. Both left-to-right and right-to-left LSTM layers contain 100 LSTM memory blocks. The learning rate is set to 1e-4, and the momentum to 0.9. The training is carried out for one million steps (roughly corresponding to 10 epochs, given the size of the training set). Training errors are averaged after every 10,000 training steps. The network corresponding to the minimum training error is used for test set evaluation.

While most of the other approaches use language modeling, font adaptation and dictionary corrections as means to improve their results, LSTM networks have shown to yield comparable results without employing these techniques. Therefore, it should be recognized that the reported results are obtained without the aid of any of the above-mentioned post-processing steps. Moreover, it should also be noted that no handcrafted features are used for the training of LSTM networks to recognize multilingual text.

8.2.5 Results

The experimental results are listed in Table 8.1, and some sample outputs are presented in Table 8.2. Since, there are no *umlauts* (German) and *accented* (French) letters in English, therefore, the words containing those special characters are omitted from the recognition results while testing LSTM model trained for German on French and model trained for English on French and German. If such words are not removed, then the resulting errors would also contain a proportion of errors due to erroneous recognition of characters that were not present in the training of the LSTM model for that language. By removing them, the true performance of the LSTM network trained

Table 8.2: LSTM network trained on English is unable to recognize special characters of German and French as they were not part of training. However, other characters of words containing these special characters are recognized correctly. Thus we can train an LSTM model for mix-data of a family of script and can use it to recognize individual language of this family with very low recognition error.

Text-line Image	*But even then the morning cock crew loud,*	let me and Mr. Collins have a little conversation together."
English	But even then the morning cock crew loud,	let me and Mr. Collins have a little conversation together.''
German	But even then the morning cock rew loud,	let me and Mr. Collins have a little conversation together.''
French	Eui even then the morning coc crev loud,	let me and Mr. Collins have a little conversation together.''
Mixed-Data	But even then the morning cock crew loud,	let me and Mr. Collins have a little conversation together.''
Text-line Image	*Aus der ich selbst zuweilen nasche,*	*Und bist du kühn und hälst du Stich,*
English	Aus der ich selbst zuweilen nasche,	LInd bist du kiihn und hlilst du Stich,
German	Aus der ich selbst zuweilen nasche,	Und bist du kühn und hälst du Stich,
French	Aus der ich selbst zutveilen nasche,	LUnd bist du kiihn und hâlst du Stich,
Mixed-Data	Aus der ich selbst zuweilen nasche,	Und bist du kühn und hädlst du Stich,
Text-line Image	*pour peu que ces livres nouveaux fussent*	avait approché ses deux mains pleines de ma lanterne
English	pour peu que ces livres nouveaux fussent	avait approche ses deux mains pleines de ma lanterne
German	pour peu que ces livres nonveaux fussent	avait approché ses deux mains pleines de ma lanterne
French	pour peu que ces livres nouveaux fussent	avait approché ses deux mains pleines de ma lanterne
Mixed-Data	pour peu que ces lives nouveaux fiussent	avait approché ses deux mains pleines de ma lanterne... Je me penchai

for language containing lesser alphabets on the language containing more alphabets can be evaluated.

LSTM model trained for Mixed-Data is able to obtain similar recognition results (around 1% recognition error) when applied to English, German and French script individually. Other results indicate little language dependence in that LSTM models trained for a single language yielded lower error rates when tested on the same script than when they were evaluated on other scripts.

To gauge the magnitude of affect of language modelling, we have compared our results with Tesseract (Version 3.02) open-source OCR system [Smi07]. The available models for English, French and German languages have been evaluated on the same test-data. Tesseract system yields very high error rates (CER) as compared to LSTM models. It seems that Tesseract models are not trained on certain fonts, thereby resulting in more recognition errors on these fonts. Tesseract OCR model for English yields 7.7%, 9.1% and 8.3% CER when applied to French, German and Mixed-Data respectively. OCR model for French returns 7.14%, 7.3% and 6.8% CER when applied to

English, German and Mixed-Data respectively, while OCR model for German returns 7.2%, 8.59% and 7.4% recognition error when applied to English, French and Mixed-Data respectively. These results show that the absence of language modeling or applying different language models in Tesseract affects the recognition poorly. Since no model for Mixed-Data is available for Tesseract, the effect of evaluating such a model on individual script could not be computed.

8.2.6 Error Analysis

The results reported in this work demonstrate that the LSTM networks can be used for MOCR. LSTM networks do not learn a particular language model internally (nor we need any such model as a post-processing step). Moreover, they show great promise to learn various shapes of a certain character in different fonts and under degradation (as evident from our highly versatile data). The language dependence is observable, but the affects are small as compared to other contemporary OCR methodologies, where absence of language models results in very bad results. To gauge the language dependence more precisely, one can evaluate the performance of LSTM networks by training them on randomly generated data using n-gram statistics and testing those models on natural languages.

In the following text, we will analyze the errors produced by the LSTM networks when applied to other scripts. Top 5 confusions for each case are tabulated in Table 8.3. The case of applying an LSTM network to the same language for which it is trained is not discussed here as it is not relevant for the discussion of cross-language performance of LSTM networks.

Most of the errors caused by LSTM network trained on **Mixed-Data** are due to its failure in recognizing certain characters like 'l,t,r,i'. These errors may be removed by increasing the training data, that contains these characters in sufficient amount.

Looking at the first column of Table 8.3 (Applying LSTM network trained for **English** on other 3 scripts), most of the errors are due to the confusion between characters of similar shapes, like 'I' to 'l' (and vice versa), 'Z' to '2' and 'c' to 'e'. Two confusions namely 'Z' with 'A' and 'Z' with 'L' are interesting as, apparently, there are no shape similarity between them. However, if the 'Z' gets noisy due to scanning artifacts, then it may look similar to a 'L'. Another possibility of this error may be due to the fact that 'Z' is the least frequent letter in English[4] and thus there may be not many 'Zs'

---

[4] http://en.wikipedia.org/wiki/Letter_frequency

Table 8.3: Top confusions when applying LSTM models for various tasks. The confusions for an LSTM models for which it was trained are not mentioned as it is unnecessary for our present work. A ”_” shows the garbage class, i.e. the character is not recognized at all. When the LSTM network trained on English is applied to recognize other scripts, the resulting top errors are similar – shape confusions between characters. Non-recognition of “*space*” and “ ′ ” are other noticeable errors. For network trained on German language, most errors are due to non-recognition of characters. Confusion of *w/W* with *v/V* are the top confusions when LSTM network trained on French is applied to other scripts.

Model **Script**	**English**	**German**	**French**	**Mixed**
English	-	$\_ \leftarrow space$ $\_ \leftarrow c$ $\_ \leftarrow t$ $\_ \leftarrow '$ $v \leftarrow y$	$v \leftarrow w$ $vv \leftarrow w\_$ $\_ \leftarrow space$ $\_ \leftarrow w$ $l \leftarrow I$	$\_ \leftarrow space$ $\_ \leftarrow t$ $\_ \leftarrow '$ $l \leftarrow I$ $\_ \leftarrow l$
German	$l \leftarrow I$ $L \leftarrow Z$ $A \leftarrow Z$ $c \leftarrow e$ $2 \leftarrow Z$	-	$v \leftarrow w$ $\hat{u} \leftarrow \ddot{u}$ $V \leftarrow W$ $\_ \leftarrow space$ $vv \leftarrow w\_$	$\_ \leftarrow space$ $\_ \leftarrow t$ $\_ \leftarrow l$ $\_ \leftarrow i$ $\_ \leftarrow r$
French	$\_ \leftarrow '$ $\_ \leftarrow space$ $I \leftarrow l$ $t \leftarrow l$ $I \leftarrow !$	$\_ \leftarrow space$ $\_ \leftarrow '$ $e \leftarrow \_$ $\_ \leftarrow c$ $\_ \leftarrow l$	-	$\_ \leftarrow space$ $\_ \leftarrow i$ $e \leftarrow \acute{e}$ $\_ \leftarrow l$ $\_ \leftarrow '$
Mixed-script	$\_ \leftarrow '$ $l \leftarrow I$ $I \leftarrow l$ $\_ \leftarrow space$ $t \leftarrow l$	$\_ \leftarrow space$ $\_ \leftarrow '$ $g \leftarrow q$ $e \leftarrow \_$ $T\_ \leftarrow l'$	$v \leftarrow w$ $\hat{o} \leftarrow \ddot{o}$ $\hat{a} \leftarrow \ddot{a}$ $V \leftarrow W$ $\hat{u} \leftarrow \ddot{u}$	-

in the training samples, thereby resulting in its poor recognition. Two other noticeable errors (also in other models) are unrecognized ‘space’ and “ ’ ” (apostrophe). An *underscore* (”_”) denotes a garbage class.

For LSTM networks trained on **German** language (second column in Table 8.3), most of the top errors are due to the inability of LSTM to recognize a particular character. Top errors when applying LSTM network trained for **French** language on other scripts are shape-confusion between *w/W* with *v/V*. An interesting observation, which could be a possible reason for such behaviour, is that relative frequency of ‘v’ is higher than ‘w’ (see previous footnote) in German and English, while it is smaller in French. So, this is a language dependent issue, which is not observable in case of Mixed-Data.

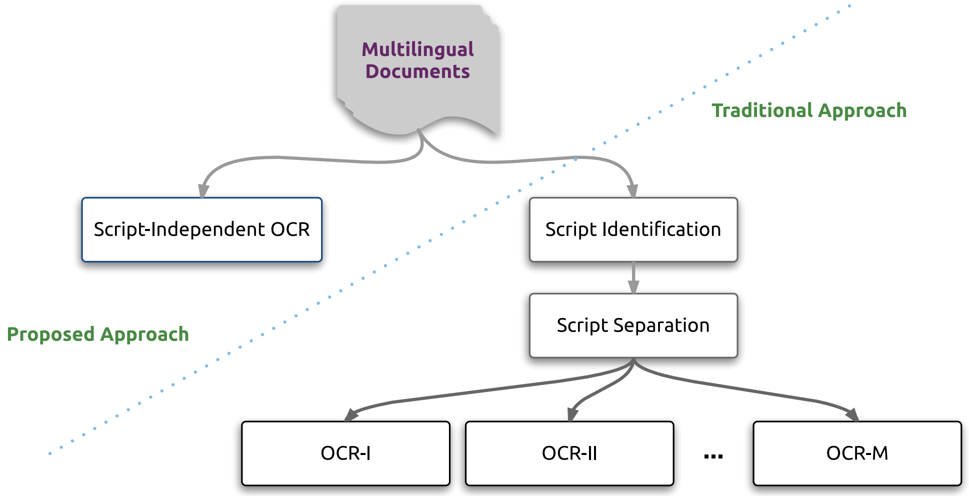


Figure 8.2: The two different approaches to OCR multilingual documents. In the traditional approach, shown on right, script identification and separation is employed so that single script OCR models can be applied. On the other hand, the proposed generalized approach directly OCR the multilingual documents.

8.2.7 Conclusion

The application of LSTM networks for language independent OCR demonstrates that these networks are capable of learning many character shapes simultaneously. Therefore, they can be utilized to recognize multiple scripts simultaneously. The next section reports a generalized OCR framework in which LSTM networks have been used to recognize multiscript documents without the aid of a separate script identification module. LSTM networks have demonstrated great ability to be used as a universal OCR engine to recognize the text in multiple languages and scripts.

8.3 Generalized OCR with LSTM Networks

Generalized OCR is the term used for an OCR system that can recognize text in multiple scripts and languages simultaneously. Encouraged by the promising OCR results obtained by the LSTM networks on language independent OCR results for Latin script, this section reports the extension of the same idea to recognize text comprising multiple scripts such that the traditionally employed script identification step can be avoided (see Figure 8.2).

The proposed methodology for generalized OCR is essentially the same as that of using LSTM networks for a single script or unilingual OCR. The sequence learning methodology for a single script or unilingual OCR system, employs the training of LSTM networks on a large corpus of text-line images whose GT information is given. The GT information contains the character labels or equivalent encoding of a single script. An LSTM network is trained to learn the sequence-to-sequence mapping between the given text-line image and associated ground-truth sequence.

In the proposed technique to OCR multilingual documents, the GT data contains the class labels representing the character-set of all scripts. LSTM networks have been used as a sequence learner on text-line images where the target labels are alphabets of multiple scripts. LSTM-based line recognizer learns the sequence-to-sequence matching between the multiscript target sequence with any given text-line image.

Salient features of the proposed approach are as follows:

- No handcrafted features are used; instead the LSTM network learns the features from the raw pixel values of the text-line images.
- No post processing has been done to correct the OCR errors using language modelling, dictionary correction or other such operations.
- Text is recognized at text-line level; thereby requiring only text-lines to be extracted from the layout step.

Our hypothesis in this work is that the LSTM networks can be utilized to OCR multiple scripts using the single OCR model. This hypothesis is based on the results reported in the literature on the usage of LSTM networks for various sequence learning tasks. To justify our hypothesis, a single LSTM-based line recognizer is trained with a corpus containing multiple scripts (in our case, this corpus contains Latin and Greek scripts).

To gauge the accuracy, standard metric of *Levenshtein* Distance is used (see Equation 5.1). The accuracy is measured at character level and reported as error rate. Experimental evaluation of the LSTM-based solution for generalized OCR are presented in the following section.

8.3.1 Preprocessing

As mentioned earlier, *normalization* is an important preprocessing step in applying 1D-LSTM networks. The filter-based normalization method, as explained in Appendix B, is used for the experiments reported in this section. This method of text-line normalization is script independent and has shown to work for both printed and handwritten text [YSBS15].

8.3.2 Database

One of the main issues in developing the MOCR technology is the unavailability of standardized databases. A lot of work has been reported for script identification on multilingual documents; however, as mentioned previously, the dataset used therein is either private or no longer available. Therefore, to evaluate our hypothesis about the generalized OCR, synthetically generated multilingual text-lines are used[5]. Though one can not replace the effectiveness of real data for training, LSTM networks, however have shown the capacity to tolerate small variations in shape images.

In order to train LSTM networks, we have used 90,000 synthetically generated text-line images in *Serif* and *Sans-Serif* fonts with normal, bold, italic and bold-italic styles (see Figure 8.3 for some example images). The process to generate artificial text-line images is explained in Chapter 4.

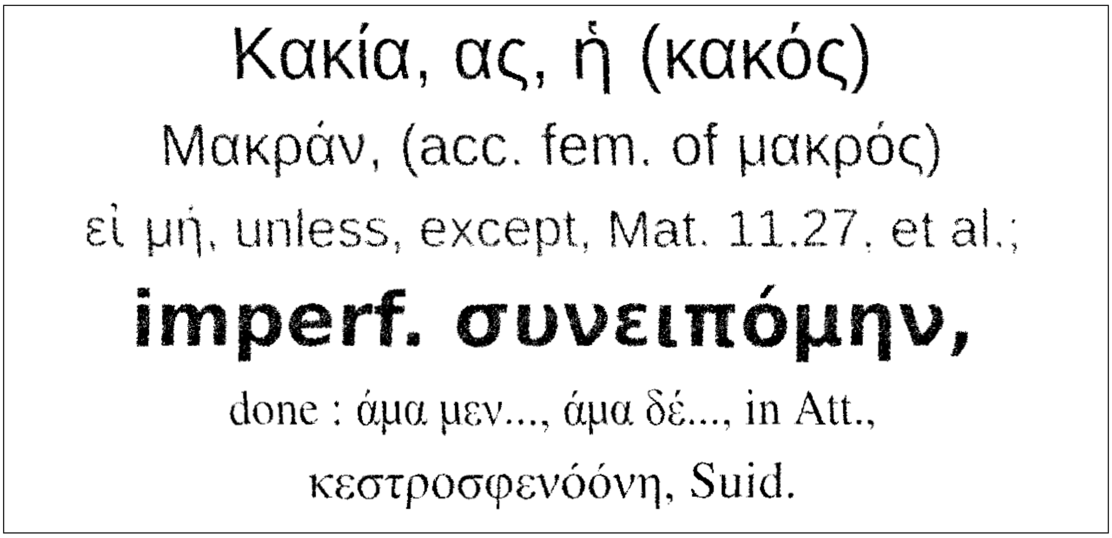
Since, these text-lines are taken from natural documents, they contain the natural variation of scripts. Some text lines contain only one script and some contain a good distribution of words from multiple scripts.

8.3.3 LSTM Architecture and Parameters

The architecture of LSTM-based line recognizer, used for Generalized OCR methodology, is shown in Figure 8.4. It is basically the same architecture that has been used throughout this thesis. The 1D-LSTM-based OCR system uses a small number of tunable parameters. One important parameter is the number of LSTM cells in the hidden layer(s) and the number of hidden layers. In this work, we used only one hidden layer with 100 LSTM memory cells in each of right-to-left and left-to-right layers (corresponding to bidirectional mode). Other parameters are learning rate (set to $1e-04$)

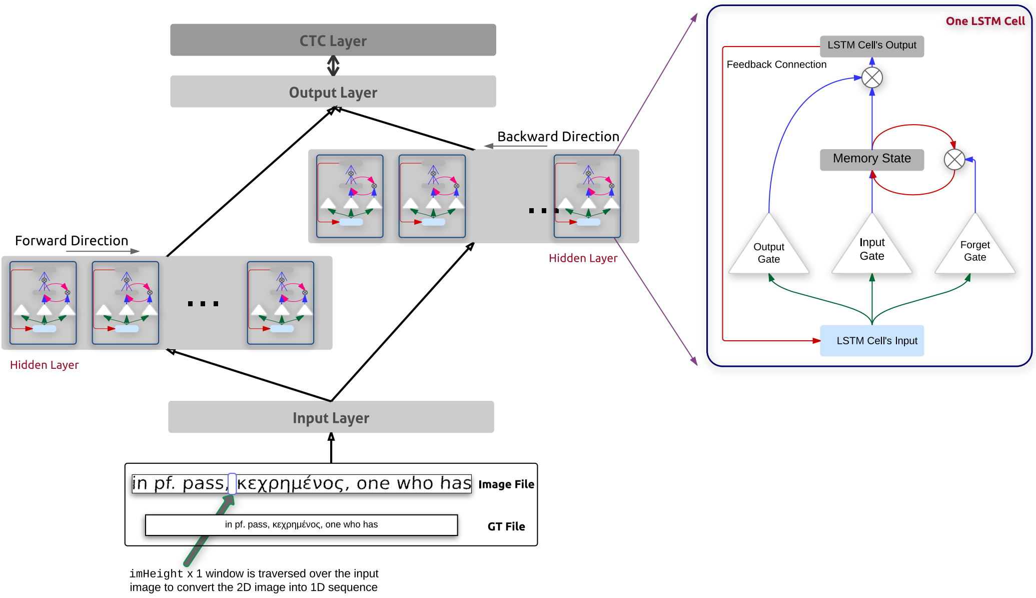
---

[5] available free-of-cost from the author



Κακία, ας, ἡ (κακός)
Μακράν, (acc. fem. of μακρός)
εἰ μή, unless, except, Mat. 11.27. et al.;
imperf. συνειπόμην,
done : ἅμα μεν..., ἅμα δέ..., in Att.,
κεστροσφενόόνη, Suid.

Figure 8.3: Some examples from the training data used for generic OCR. These images were synthetically generated using *OCRopus* open-source system [OCR15].



CTC Layer
Output Layer
Backward Direction
Forward Direction
Hidden Layer
Hidden Layer
Input Layer
in pf. pass, κεχρημένος, one who has
Image File
in pf. pass, κεχρημένος, one who has
GT File
imHeight x 1 window is traversed over the input image to convert the 2D image into 1D sequence
One LSTM Cell
LSTM Cell's Output
Feedback Connection
Memory State
Output Gate
Input Gate
Forget Gate
LSTM Cell's Input

Figure 8.4: The network architecture used for the Generalized OCR methodology. The input text-line image along with corresponding ground-truth is given at the input layer. In bidirectional mode of LSTM, one hidden layer scans the input image in forward direction while an identical layer scans in the backward direction. Both layers are connected to a single output layer. CTC layer aligns the output activation with the GT using the forward-backward algorithm. A simplified LSTM cell is also shown on the right.

and the momentum (set to 0.9) in the reported experiments. These parameters are the same as that used in [BUHAAS13], because the performance of an LSTM line recognizer is fairly unaffected by the use of other numbers.

The network has been trained for 1 million iterations (please refer to Figure 8.5 to see how training errors appear at each iteration) and the intermediate models were saved after every 10,000 iterations. This creates a total of 100 models. The training errors for these models are computed, and then the network with the least error has been selected as the best network, which is then used to OCR the test data.

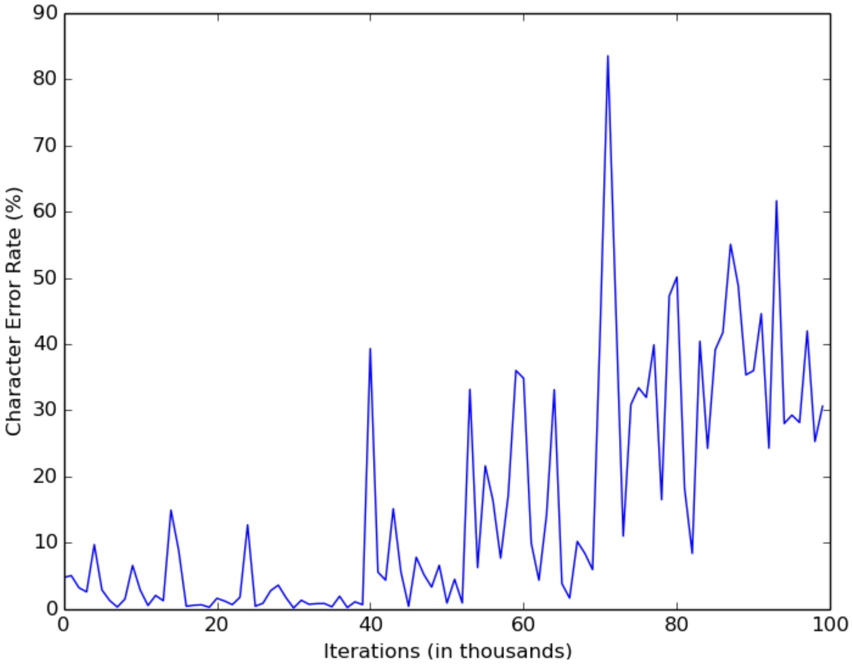


Figure 8.5: Training errors for LSTM network. The model saved with the minimum error is applied on the test dataset to compute the generalization error. It is to be noted that the network starts diverging around 40^{th} mini-batch. However, the nearly "flat" error-curve just before indicates a stable network.

8.3.4 Results

To test the generalization capability of the trained LSTM network, 9,900 synthetically generated text-line images (using the same methodology described in Section 8.3.2) are used. The LSTM-based generalized OCR model yields an error rate of 1.28% on this test data. A sample image which is correctly recognized with the best trained model is shown in Figure 8.6 and an image on which the same model fails in predicting correct labels is shown in Figure 8.7.

8.3.5 Error Analysis

Top confusions for OCR using the proposed generalized MOCR approach are tabulated in Table 8.4. It can be observed that many errors are due to the *insertion* or *deletion* of punctuation marks and 'space' *deletions*. This is understandable because of the small size of punctuation marks. Other source of errors are the confusion between

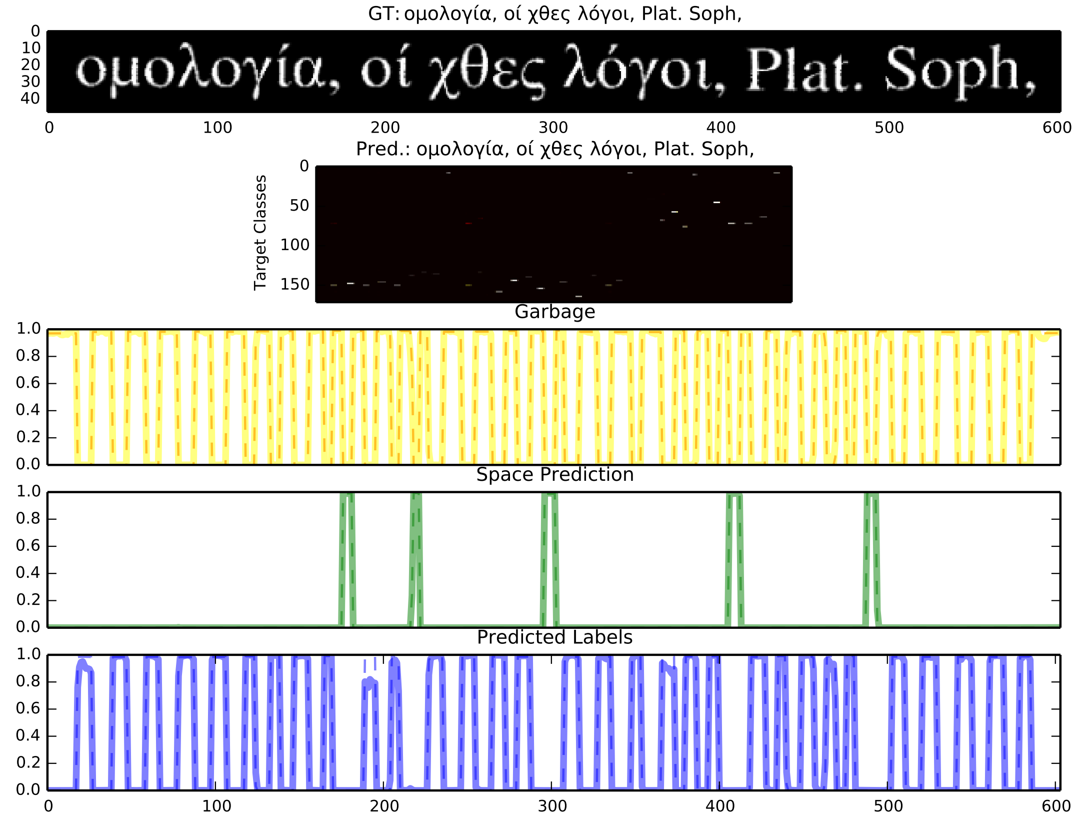


Figure 8.6: An example of correctly recognized text-line image. This figure shows a sample test image on which the best trained LSTM model predicts every label correctly. The first row shows the original image with corresponding GT. The second row shows the confusion table in the image form, with the predicted text given at the top. The vertical axis shows target classes. The third row shows the probability map of the 'Garbage' class assigned to the individual frames. The fourth row shows the probability of estimating the 'space' between the words. The fifth row shows the posterior probability of all other class labels. The dotted lines in the third, fourth, and fifth rows indicate the probability of aligned labels using CTC, while the solid lines represent the raw output activations. Note that how the probability value of predicting ο (Greek omicron) is comparatively low. However, the LSTM model has correctly predicted this label in this text-line.

the similar characters in both scripts. These letters such as o/ο, X/Χ, O/Ο (first characters are Latin, while latter ones are Greek) are even indistinguishable to human eyes. But if one knows the context, it is easier to recognize any character. However, when these characters occur in conjunction with the punctuation marks, the recognition of a character becomes difficult because in this case context would not help LSTM networks to recognize. To elaborate this point further, consider Table 8.5 where confusions are shown considering neighboring context. There are many instances of similar characters accompanied with punctuation marks. These pairs makes the contextual processing of neural network difficult, resulting in substitution errors. However, it must be noted that these errors are due to the similarity of some characters in both scripts. Both scripts (English and Greek) share some characters' shape (around 16),

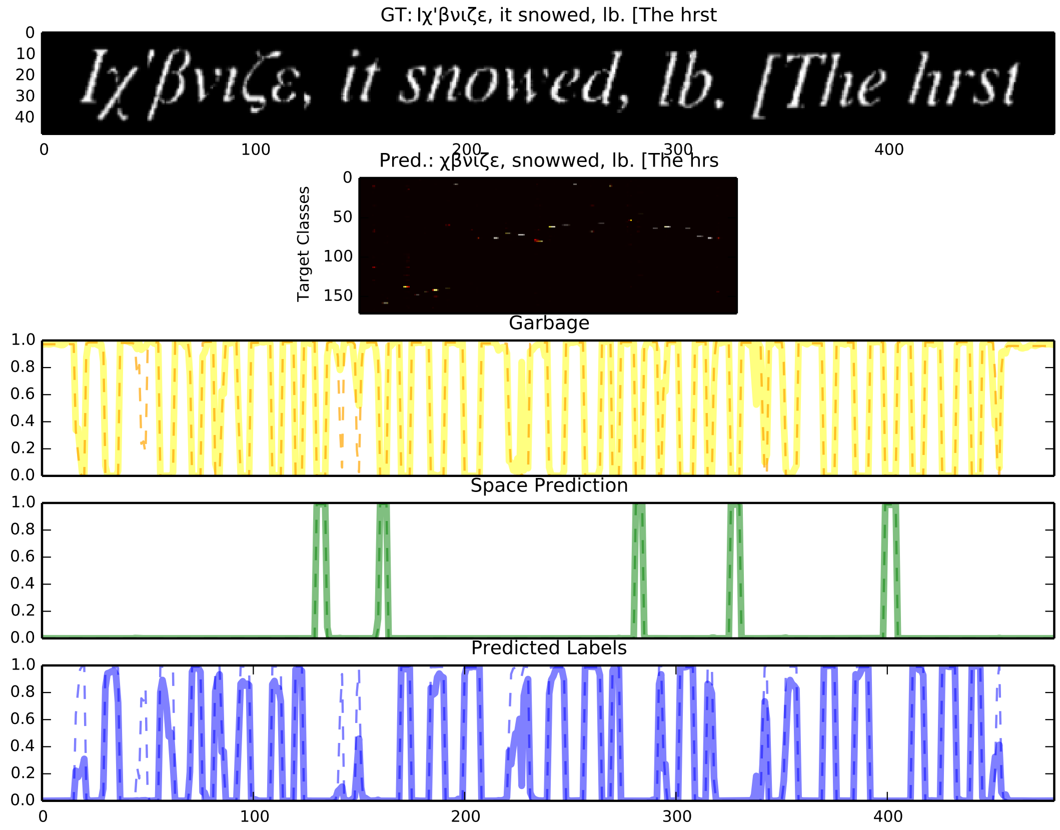


Figure 8.7: A sample text-line image with incorrectly predicted labels. This figure shows a sample test image on which the LSTM model yields 7 errors. The meaning of each row is the same as described in Figure 8.6. The first error is the deletion of " ' ". LSTM network erroneously deletes this label and predicts a 'space' instead. Other characters that are deleted include 'i','w' and 'l'. The corresponding confusion matrix (second row) shows that LSTM model is confused between many labels (as indicated by multiple dots in a single frame). The probability map (third row) shows that characters that are either deleted or substituted have low probability scores, resulting in bad OCR result.

resulting in substitution errors. These errors will not be present in case of scripts that are markedly different in terms of their grapheme structure, e.g., English and Devanagari or Latin and Arabic.

8.4 Chapter Summary

This chapter validates the capability of LSTM networks for the OCR of multilingual (with multiple languages and scripts) documents. As a first step, a single LSTM model is trained with a mixture of three European languages of a single script, namely, English, German and French. The OCR model thus obtained produces very low errors

Table 8.4: Top 15 confusions from generalized OCR methodology. **GT** – Ground-Truth, **Pred.** – predicted and **No.** – No. of occurrences. '_' denotes a deletion and blank cell represents a space.

Total No. of Characters	295096
Total No. of Errors	3770
Character Error Rate (CER) (%)	1.28%

Pred.	**GT**	**No.**
	_	122
l	I	71
_	.	61
	_	59
_	*l*	54
X	X	52
o	o	51
c	e	50
_	I	47
O	O	43
.	_	42
′A	X	41
o	o	38
,	_	37
l	_	34

Table 8.5: Top confusions from generalized OCR methodology considering the neighboring context. **GT** – Ground-truth, **Pred.** – predicted and **No.** – No. of occurrences. '_' denotes a deletion and blank cell represents a space.

GT	**Pred.**	**No.**	**GT**	**Pred.**	**No.**	**GT**	**Pred.**	**No.**
_i	li	29	ς,	ς_	24	:	;	20
lli	l_i	19	a.	a_	19	O.	O.	18
H.	H.	18	′Aρ	Xρ	18	P.	P_	16
I_.	II.	16	Ib	lb	13	lo	Io	13

in recognizing the text of these languages without using any post-processing techniques, such as language modeling or dictionary correction. The language dependence is observed by reduction in the character recognition accuracy as compared to the single language OCR; however, the effect is small in comparison to other OCR methods.

The idea behind the language independent OCR is then extended to a generalized OCR framework that can OCR multilingual documents comprising multiple scripts. The presented methodology, by design, is very similar to that of a single script OCR system and does not employ the traditional script identification module. A single LSTM-based OCR model is trained for Latin-Greek bilingual documents that yields very low Character Error Rate (CER) on a dataset consisting of 9,900 text-lines.

The results of our experiments underpin our claims that multiscript OCR can be done without doing separate script identification step, thereby, we can save the efforts spent on script identification and separation. The proposed system can be retrained for any new script by just specifying the character-set of that script during the training phase. Secondly, the OCR errors are mainly due to the similarity of both scripts. The algorithm could be tested further on other multilingual documents containing other languages and scripts, such as, Arabic and English, Devanagari and English, and many more.

Chapter 9

Conclusions and Future Work

This thesis contributes in the field of Optical Character Recognition (OCR) for printed documents by extending the use of contemporary Recurrent Neural Networks (RNN) in this domain. There has been an increasing demand in developing reliable OCR systems for complex modern scripts like Devanagari and Arabic, whose combined user population exceeds 500 million people around the globe. Likewise, large scale digitization efforts for historical documents require robust OCR systems to preserve the literary heritage of our world. Furthermore, an abundance of multilingual documents present today in various forms intensifies the need of having usable OCR systems that can handle multiple scripts and languages.

This thesis contributes in two ways to solve some of these issues. Firstly, several datasets have been proposed to evaluate the performance of OCR systems for printed Devanagari and Polytonic Greek scripts. Databases for OCR tasks related to multilingual documents, such as script identification and cross-languages performance of an OCR system, have also been proposed. Additionally, a bilingual database to evaluate script independent OCR system has been developed.

Secondly, Long Short-Term Memory (LSTM)-based OCR methodology has been assessed for some modern scripts including English, Devanagari, and Nastaleeq. This methodology has also been evaluated for some historical scripts including Fraktur, Polytonic Greek, and medieval Latin script of 15th century. A generalized OCR framework for documents containing multiple languages and scripts has also been put forward. This framework allows the use of a single OCR model for multiple scripts and languages.

Several conclusions can be drawn from the work done in this thesis.

- The first and the foremost is that the use of artificial data, if generated carefully to reflect closely the degradations in the scanning process, can replace the need for a large collection of real ground-truthed datasets. Experiments performed by training the LSTM-based line recognizer on synthetic data and testing it on real scanned documents justifies this claim.
- The marriage of segmentation-based and segmentation-free approaches to OCR documents, for which GT data is not available, results in a framework that can self-correct the ground-truthed data in an iterative manner.
- The powerful context-aware processing makes the LSTM-based OCR network a suitable sequence learning machine that can perform on mono-lingual scripts, as well as on multiple scripts. The use of 1D-LSTM networks requires very few parameters to tune and these networks outperform more complex MDLSTM networks, if the input is normalized properly. Moreover, the performance is better when features are learned automatically instead of when handcrafted features are used.

There are multiple directions in which the work reported in this thesis can be further extended. Some of the key future directions are listed below:

- The application of LSTM-based OCR methodology can be directly extended for the case of camera-captured documents. The challenge in camera-captured documents is the presence of curved text-lines. There are several techniques to correct this issue, including image dewarping or text-line extraction directly from the camera-capture documents. If these issues are taken care of, the application of LSTM-based OCR is straight forward and simple.
- The Hierarchical Subsampling LSTM (HSLSTM) networks have shown excellent results on Urdu Nastaleeq OCR. However, their performance could be more thoroughly tested on other scripts to better gauge their potential.
- The LSTM-based OCR reported for Urdu Nastaleeq can be extended further to other similar scripts, e.g., Persian, Pushto, Sindhi, and Kashmiri.
- The OCR of Devanagari can be improved by employing a mechanism that addresses the issue of vertically stacked characters. MDLSTM networks may yield better results, or alternatively, improved preprocessing may benefit 1D-LSTM networks.
- The generalized OCR framework is presently evaluated for two only scripts (Greek and English). If more datasets containing multiple scripts emerge, this framework can be used to further establish its performance.

It is hoped that the work in this thesis successfully attempts to fulfill the gap that was present in the field of OCR for printed documents, and would serve as a stepping stone in future research endeavors.

Appendix A

Long Short-Term Memory Networks

This appendix details the internal working of Long Short-Term Memory (LSTM) networks. The first part of this appendix overviews the necessary mathematical details about the LSTM network. The second part discusses various architectures of LSTM networks that have been used commonly for various tasks, including printed OCR. The last part describes various parameters that one needs to tune when using the LSTM networks.

A.1 LSTM Networks

LSTM networks are a recent architecture of Recurrent Neural Networks (RNN). RNNs are similar to feedforward neural networks with the exception that neurons at hidden layers are self-connected. These connections form directed graphs in the architecture [Gra12]. A typical recurrent neural network is shown in Figure A.2.

The feedback mechanism enables RNNs to remember inputs at different timesteps. They are therefore, very good sequence learners. However, in practice their applications were quite limited till the late 1990s. This was mainly because of the training, where typical gradient descent based algorithms (first order derivatives) generally fail to converge or take too much time [BSF94]. Recently proposed 2^{nd} order derivative based methods however have shown to train RNNs much more efficiently [Sut12]. Another issue with early RNNs is the so-called “Exploding/Vanishing Gradient Problem”, which implies that the gradients, during the training, either become very large or very small, thus resulting in poor training [HS97].

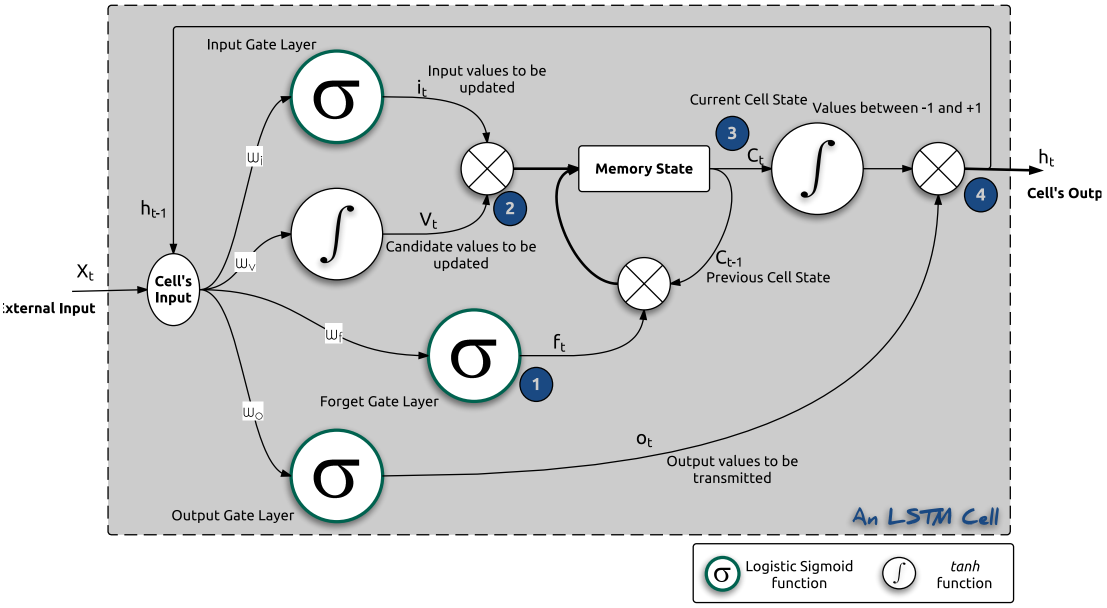

Input Gate Layer
Input values to be updated
i_t
Current Cell State
Values between -1 and +1
3
Memory State
C_t
h_t
Cell's Outp
4
w_i
h_{t-1}
V_t
2
Candidate values to be updated
X_t
w_v
Cell's Input
xternal Input
C_{t-1}
Previous Cell State
w_f
f_t
1
Forget Gate Layer
w_o
o_t
Output values to be transmitted
Output Gate Layer
An LSTM Cell
Logistic Sigmoid function
tanh function

Figure A.1: Illustration of a basic LSTM memory cell. The *cell input* at any time step consists of external input and the cell state from previous timestep. At each timestep, the cell has to decide which information it needs to update and which information it needs to forget. The information flow inside the cell is indicated by numbered circles. The first step is to decide about the information that is no longer required. The logistic sigmoid layer, also called *Forget Gate Layer*, makes this decision by applying a sigmoid function to the cell input and produces a value between '0' (do not retain at all) and '1' (retain everything). The second step decides which values are going to be updated in the *Cell Memory*, and it consists of two sub-steps. Firstly, a sigmoid layer, called *Input Gate Layer* selects the values from the cell input, and secondly, a *tanh* layer creates a candidate vector of input values that could be added to the Memory State. These two stages are combined through a multiplicative gate, shown by $\otimes$. With this information available, the cell updates the current memory state (step 3) by combining linearly the output from 1^{st} and 2^{nd} steps. Finally, another sigmoid layer, termed as *Output Gate Layer* decides about the output value. However, before deciding on the values, the Memory State is filtered through another *tanh* layer so that the output values range between -1 to $+1$. All the gates shown have a bias input in addition to the inputs shown in this figure.

Hochreiter and Schmidhuber [HS97] proposed to change the basic unit of RNN, that is a simple neuron with a computer memory-like cell, called LSTM cell. A simplified LSTM cell is shown in Figure A.1.

The basic idea is to update the memory cell state represented as "**Memory State**" in Figure A.1 or also termed in the literature as "**Cell State**". To update the Cell State, two inputs are important: 1) the information that is no longer required (so that the cell can forget about this information) and 2) new information that has to be stored in the cell state. The "**Forget Gate Layer**" decides about the retention of the input.

It is a sigmoid function given by

$$\sigma(x) = \frac{1}{1 + exp(-x)} \tag{A.1}$$

The sigmoid function ranges between '0' and '1'. The forget layer determines inputs that are no longer needed to be retained. Following Equation A.2 shows this process mathematically.

$$f_t = \sigma(W_{xf}.x_t + W_{hf}.h_{t-1} + b_f) \tag{A.2}$$

where f_t is the forget gate's output. W_{xf} denotes the connection strength (weight of the connection) between the external input x_t and the forget gate, W_{hf} denotes the weight of the connection between the previous hidden state and the forget gate, and b_f denotes the bias of the forget gate.

Information that needs to be retained in the *cell state* is determined separately (denoted by (2) in Figure A.1). It consists of two parts: firstly, the "**Input Gate Layer**" decides which inputs are to be retained by applying a sigmoid function similar to the forget gate. Equation A.3 formulates this process mathematically. Secondly, a ***tanh*** layer provides a candidate list (v_t) (as shown by Equation A.4) that could be added to the cell state. Both of these instructions are combined by a multiplicative gate (denoted by $\otimes$).

$$i_t = \sigma(W_{xi}.x_t + W_{hi}.h_{t-1} + b_i) \tag{A.3}$$

where W_{xi} denotes the weight between the external input and the input gate, W_{hi} denotes the weight between the previous hidden state and the input gate and b_i is the bias of that unit.

$$v_t = tanh(W_{xv}.x_t + W_{hv}.h_{t-1} + b_v) \tag{A.4}$$

where *tanh* is the tangent hyperbolic function defined as,

$$tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

The *cell state* C_t is now updated with this information obtained from previous two steps.

$$C_t = f_t * C_{t-1} + i_t * v_t \tag{A.5}$$

where C_{t-1} is the previous cell state.

The final step is to produce the cell output. The "**Output Gate Layer**" is another sigmoid function that selects cell inputs that are to be carried forward (as shown by the following Equation A.6).

$$o_t = \sigma(W_{xo}.x_t + W_{ho}.h_{t-1} + b_o) \tag{A.6}$$

where W_{xo} is the weight strength between the external input x_t and the output gate, W_{ho} is the weight of connection between the previous hidden state and the output gate and b_o is the bias of this unit. The *cell state* (h_t) is passed through a *tanh* layer to push the values between the range -1 and +1. The cell output is finally calculated through Equation A.7.

$$h_t = o_t * tanh(C_t) \tag{A.7}$$

Apart from a different processing unit, an LSTM memory cell, and feedback at hidden layer, the usage of LSTM networks is very much like multilayer perceptron or any other recurrent neural network architecture. The input is given to the network at input layer. The output generated during the *forward* pass (following the Equations A.2 to A.7) is then compared to the target and the error is then back-propagated through the hidden layers to update weights of the neural network's internal connections to reduce the error between the actual output and the target output. There could be multiple error metrics like *cross-entropy* or *least square error*, etc.

In its core, the back propagation is the repeated application of chain rule to calculate the derivative of error function with respect to the network's weights. In feedforward networks, these derivatives can be calculated by using the error derivatives from the layer above. This is not the case in recurrent neural networks, where self-loops would require to express the error derivative in terms of itself. The backpropagation through time (BPTT) [Wer90] was designed to overcome this issue by "unrolling" the recurrent connections to transform the RNN into a deep feedforward network.

A.1.1 Connectionist Temporal Classification (CTC)

Any neural network requires a well-defined input-output pair. This means that the input has to be segmented before hand for every target value. For text recognition, this requires individual character segmentation. This pre-segmentation is, however, very difficult in many scenarios, such as in complex scripts, cursive writing, handwritten documents or printed documents where characters are touching due to various reasons. Segmentation-based recognition strategies for OCR usually lead to inferior results. This peculiar requirement renders RNNs unattractive for several sequence learning tasks, where they could have been a natural choice. The implication of this requirement is to use some post-processing techniques to generate a sequence at the output. Many researchers tried RNN-HMM hybrid architectures to overcome this issue without much success.. Graves et al. [GFGS06] proposed the *Connectionist Temporal Classification (CTC)*, which removes this requirement of pre-segmented inputs. This algorithm is inspired by the forward-backward algorithm of HMM, and is used to align the target labels with the output activations of the neural network, thereby removing the need for pre-segmenting the inputs.

The network output activation provides the posterior probability of characters of the input string. Let the posterior probability at timestep, t, of a character, c, is given as y_{tc}. Then, at a certain timestep, the individual character probability can be given as the $max(y_{tc})$. The network is augmented with an additional class 'ϵ' to denote the blank label. So, at each timestep, either a character class is present in the output activation or a blank label, e.g., for a given input containing the string 'abc', the output activation may look like '$\epsilon\epsilon\epsilon aaaa\epsilon\epsilon bbbb\epsilon\epsilon\epsilon\epsilon cccc\epsilon\epsilon\epsilon$'. The next step is to remove the repetitive symbols and to remove the blank labels. Every possible sequence is assigned a score,

$$s(seq) = max(s(p))$$

where $s(p)$ is given in terms of output activation as the multiplication of all y_{tc}. There could be exponential possible paths and the best path is found using a procedure similar to the Viterbi algorithms [Vit67] in HMM.

To integrate the CTC algorithm into the training process of LSTM network, the training algorithm looks like the following.

1. Compute the output activations y_{tc} through the neural network.
2. Compute the probability of starting at $t = 0$ and reaching y_{tc}.
3. Compute the probability of starting at y_{tc} and arriving at the end of the text-line.
4. Compute the probability of each target symbol given an output activation.
5. Calculate the error gradient.
6. Use backpropagation to compute the error gradients for the network weights.
7. Update weights.

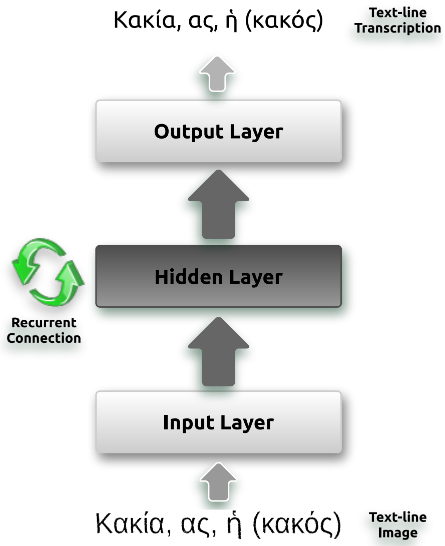

Κακία, ας, ἡ (κακός)
Text-line Transcription
Output Layer
Hidden Layer
Recurrent Connection
Input Layer
Κακία, ας, ἡ (κακός)
Text-line Image

Figure A.2: Recurrent Neural Networks (RNN) where neurons at the hidden layer form self-loops for feedback. Inspired from [Gra12].

A.2 LSTM Architectures

There are many LSTM architectures reported in the literature. It is important to distinguish between various types, so that one can choose the right type. In this Section, details about some of the commonly used LSTM networks are presented.

A.2.1 Single Layer LSTM Network

Single layer LSTM network is the simplest of LSTM networks. It consists of a single hidden layer with N LSTM cells. The last section outlines and describes the internal working of such an architecture. Figure A.2 shows this network.

A.2.2 Multilayer LSTM Network

A multilayer LSTM network is an extension of single layer network. The only difference between a single layer and multilayer network is that the later contains multiple hidden layers instead of a single layer. Figure A.3 shows an example of multilayer LSTM networks. The equations shown for a single layer LSTM network (Equation A.2–Equation A.7) remain the same except an additional summation is introduced to account for all the hidden layers.

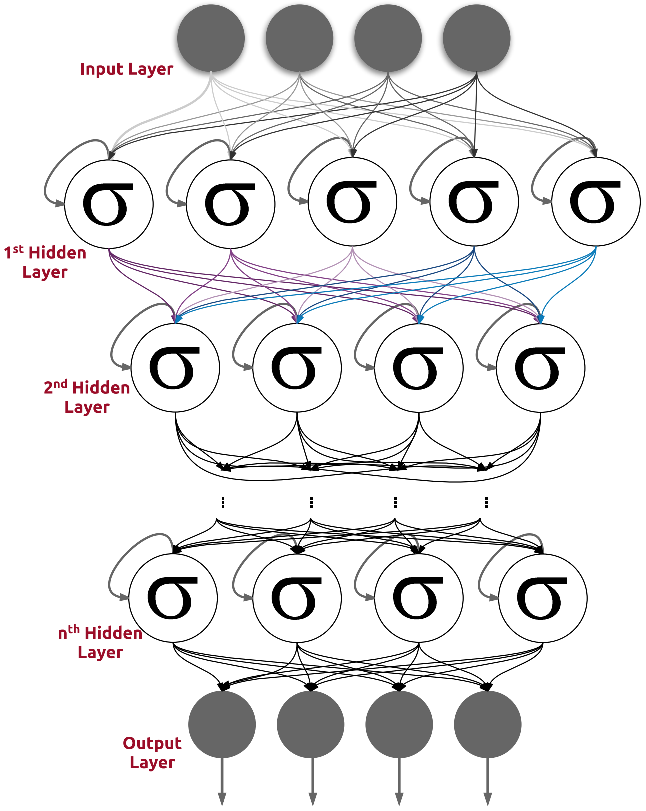


Figure A.3: A Multilayer LSTM consists of LSTM cells arranged in multiple layers. The functioning of such a network is the same as that of single layer LSTM network.

A.2.3 Hierarchical SubSampling LSTM (HSLSTM) Networks

In theory, an LSTM network with a single layer is able to learn sequence-to-sequence mapping of arbitrary length. However, in practice, it is unable to do this for very long sequences [Gra12]. A common solution is to apply subsampling on the sequence. This is done by collapsing consecutive timesteps into a single block or windows. In a multilayer setting, where output of one layer is the input of another layer, the sequence length can be reduced at each step before feeding it to the next level in the hierarchy. Graves [GS08] introduced this concept in LSTM networks for Arabic Handwriting recognition task. A schematic diagram of a hierarchical network (a network with multilayer architecture) with subsampling is shown in Figure A.4. Subsampling window or block sizes are chosen empirically depending upon the task. In order to reduce the number of weight connections between hidden layers, a feedforward layer is incorporated [Gra12].

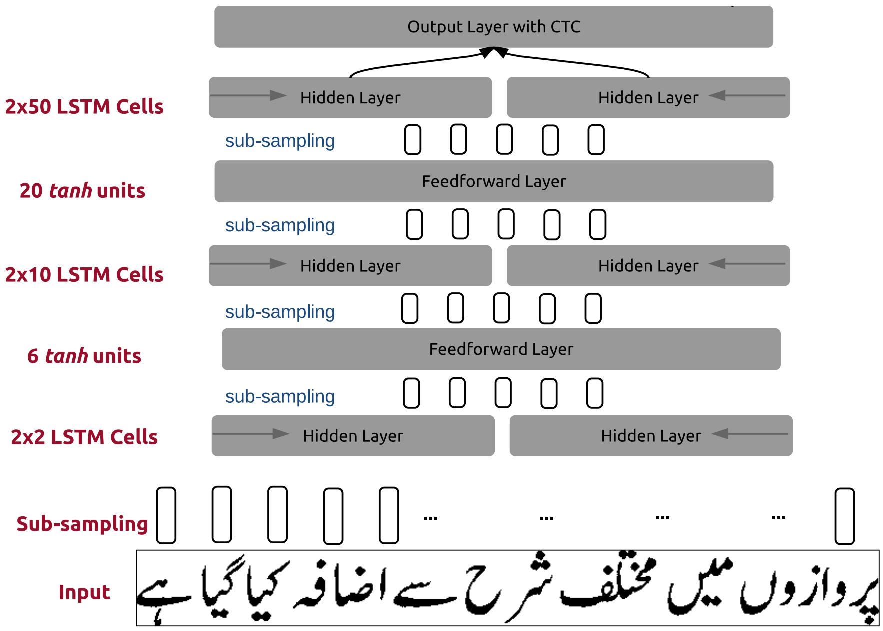

Output Layer with CTC
2x50 LSTM Cells
Hidden Layer
Hidden Layer
sub-sampling
20 *tanh* units
Feedforward Layer
sub-sampling
2x10 LSTM Cells
Hidden Layer
Hidden Layer
sub-sampling
6 *tanh* units
Feedforward Layer
sub-sampling
2x2 LSTM Cells
Hidden Layer
Hidden Layer
Sub-sampling
...
Input
پروازوں میں مختلف شرح سے اضافہ کیا گیا ہے

Figure A.4: Schematic diagram of a subsampling hierarchical network. The input from each layer is subsampled before feeding it to the next layer except the output layer. The two hidden layers show the bidirectional scanning with arrows indicating the scanning direction. The number of LSTM cells at each hidden layer is highlighted on the left (2,10,50). Two *tanh* layers are used to separate the hidden layers in order to reduce the number of connections. The size of subsampling block for 1D-LSTM networks is given as a scalar quantity instead of a window size.

A.2.4 Bidirectional LSTM Networks (BLSTM)

RNNs are very good at context-aware processing. In order to access the future context of a timestep (for offline data processing, the whole sequence is already available), Graves et al. [GS05] introduced the so-called BLSTM networks. In this type of network (which can be a single layer or multilayer with subsampling), the data sequence is processed by two identical hidden layer(s). One layer or one set of multiple layers process the data in the forward direction (left-to-right in case of a text-line) and the other identical layer or set of multiple layers process the data in the reverse direction (right-to-left in case of a text-line). Both of these layers are connected to a single output layer. Figure A.5 shows a single layer BLSTM network.

A.2.5 One Dimensional LSTM (1D-LSTM) Networks

In a One Dimensional LSTM (1D-LSTM) network, the input sequence is traversed by a sliding window of width of 1-pixel and of height equal to the image height. This

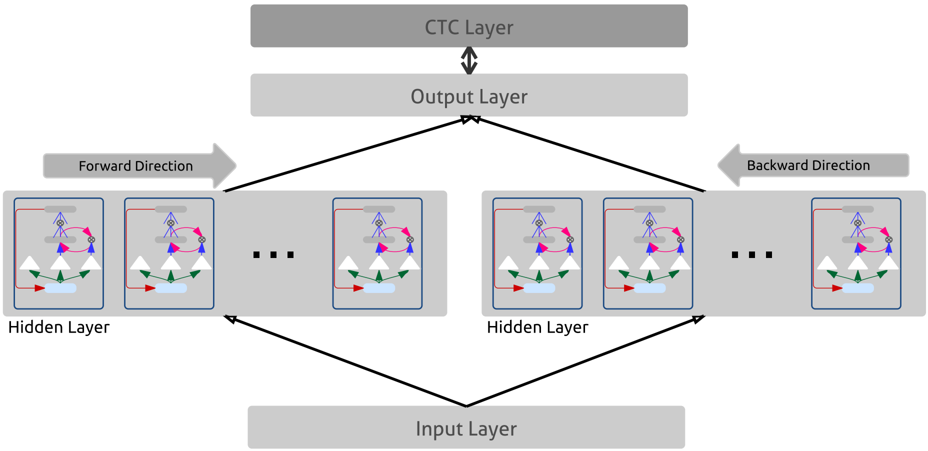


Figure A.5: The information flow in a BLSTM network. The BLSTM consists of two identical hidden layers. One layer scans the input in forward direction, while the other scans the input in reverse direction. This is done to access the context from previous and future timesteps.

converts the input sequence into a one-dimensional sequence. The height of the image is termed as the "depth" of the 1D sequence and each slice of the image thus obtained is called a "*frame*". It is, therefore, necessary to make the height of all images equal (a process termed as "normalization" in this thesis), so that the individual *frames* are of equal size. Figure A.6 depicts the process of converting the image into a one-dimensional sequence.

A.2.6 Multidimensional LSTM (MDLSTM) Networks

In a Multidimensional LSTM (MDLSTM) network of dimensions '*n*', each LSTM cell has '*n*' self-connections with '*n*' forget gates. This makes every timestep to connect to its context in all '*n*' dimensions. Moreover, instead of frame, each pixel value is presented at each timestep. The training of such networks thus takes much more time than their 1D siblings. In our experience, the performance gain is not very significant as compared to the increase in computational time. MDLSTM networks are not used in the course of this thesis work.

A.3 Tunable Parameters

There are various parameters that one needs to tune so that the LSTM networks perform satisfactorily on a given problem. In this section, tunable parameters used in various types of LSTM networks are described.

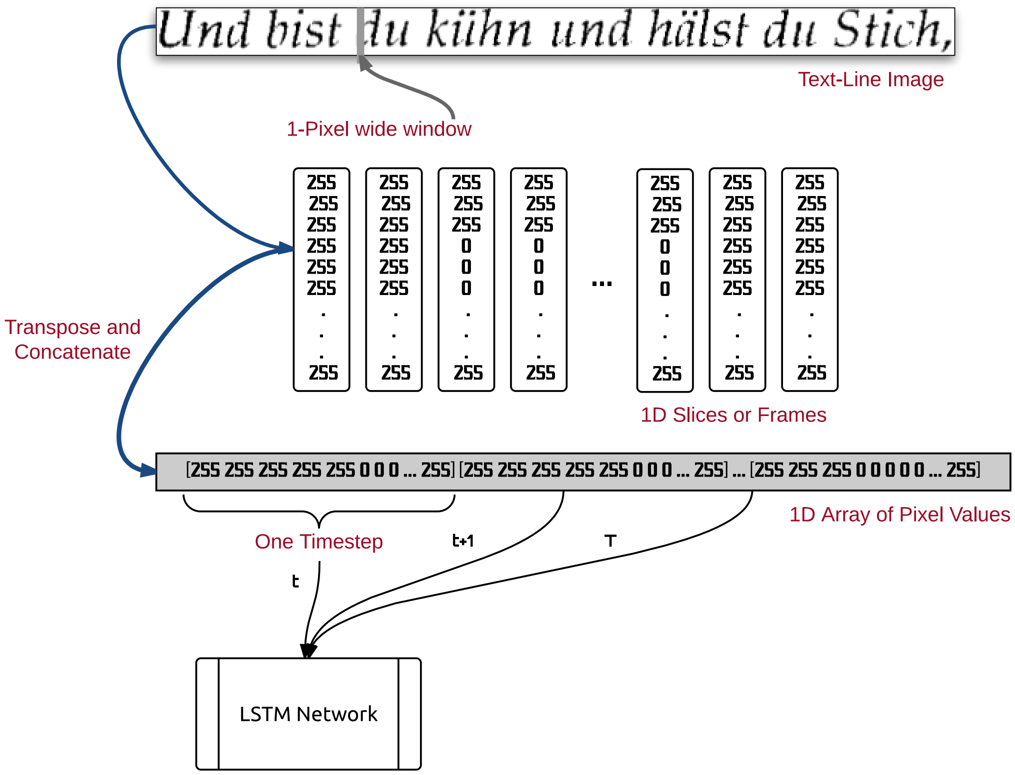

Und bist du kühn und hälst du Stich,
Text-Line Image
1-Pixel wide window
Transpose and Concatenate
1D Slices or Frames
[255 255 255 255 255 0 0 0 ... 255] [255 255 255 255 255 0 0 0 ... 255] ... [255 255 255 0 0 0 0 0 ... 255]
1D Array of Pixel Values
One Timestep
t
t+1
T
LSTM Network

Figure A.6: The process of converting a text-line image into a 1D sequence. In order to use 1D-LSTM networks, the input text-line image (a 2D sequence) is converted to a 1D sequence by traversing a 1-pixel wide window over the image in horizontal direction and then concatenating the slices thus obtained into one vector. A single slice or frame is provided at the input layer at each timestep of training. The LSTM learns to associate each frame with a class label, which also includes *reject* class.

Parameters for 1D-LSTM Networks: For 1D-LSTM networks, the most important parameter is the size of the hidden layer, that is, the number of LSTM cells in the hidden layer. Increasing the number of cells increases the performance, but it also increases the training time. See Section 5.1.1 for a discussion on how to optimally select this size. Other parameters are learning rate and momentum; but the LSTM performance is generally unaffected by changing these parameters.

Parameters for MDLSTM Networks: The number of hidden layers is also an important parameter along with the number of LSTM cells in an individual layer. Though with more layers, the learning capability of a network increases as it learns more compact representation, but it also makes the training more difficult [Gra12].

Parameters for HSLSTM Networks: These networks need a lot of parameters to be considered. Apart from the number of layers and the number of cells in a layer, it is also important to select the optimal sizes for various subsampling window sizes. The relative sizes of hidden layers are also important in this type of architecture. The number of cells in hidden layers at the top of hierarchy should be far more than those in lower level to ensure that the total time does not exceed than twice of that consumed at the lowest layer.

Appendix B

Text-Line Normalization

The relative position and scale of individual characters in a text-line are important features for Latin and many other scripts. *Normalization* of text-lines helps in making this information consistent across all text-lines in a given database. There are many normalization methods proposed in the literature. Normalization methods that have been used for various experiments reported in this thesis are described in the sections below.

B.1 Image Rescaling

Image rescaling is the simplest method to make the heights of all images in a database equal. For a desired image height, a scale can be calculated as following:

$$scale = \frac{target\_height}{actual\_height}$$

This scale is then used to determine the width of the "normalized" image by simply multiplying it with the width of the actual image.

$$target\_width = scale * actual\_width$$

This normalization is used in the current thesis for some of the OCR experiments reported for Urdu Nastaleeq script.

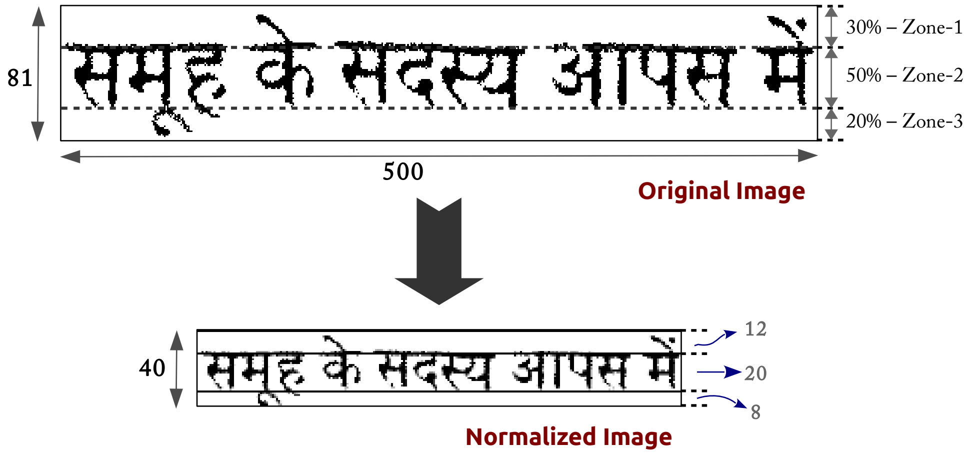


Figure B.1: Zone-based normalization for Devanagari text-line image. Three zones are shown with corresponding percentages of the image. In the bottom row, a normalized image is shown with a height of 40 pixels.

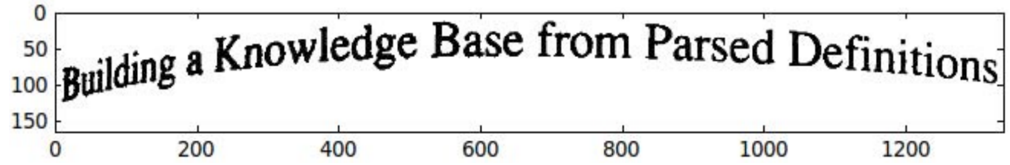
B.2 Zone-Based Normalization

Characters in many scripts like Latin, Greek and Devanagari follow certain typographic rules. A text-line in such scripts can be divided into three zones. A *baseline* passes through the bottom of majority of the characters, and a *mean-line* is at the middle height from the baseline to the top edge of a text-line. Most of the small characters, like ‘x’, ‘s’, and ‘o’ lie between these two lines. The portion of the characters that extends above the mean-line is termed as ‘ascender’, and that extending below the baseline is termed as the ‘descender’. The zone between the baseline and the mean-line is the *middle-zone*, the zone below the mean-line is the *bottom-zone* and the zone above the baseline is called the *top-zone*. A sample text-line in Devanagari script with these three zones is shown in Figure B.1.

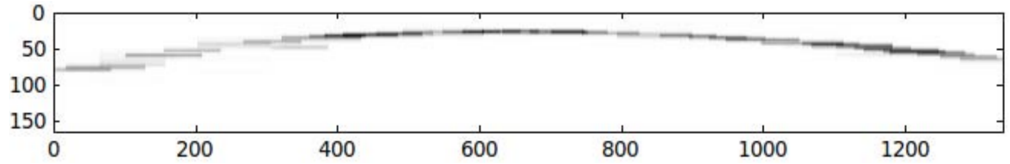
Rashid et al. [Ras14] proposed a text-line normalization method which uses the above-mentioned three zones. Statistical analysis is carried out to estimate these zones in an image and then each zone is rescaled to a specific height by simple rescaling described in the previous section. This normalization method has been employed for the experiments reported for Devanagari script in this thesis.

B.3 Token-Dictionary based Normalization

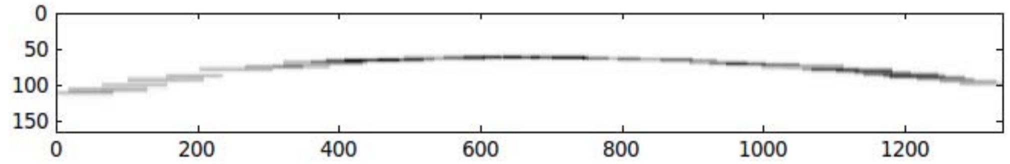
This text-line normalization method is based on a dictionary composed of connected component shapes and associated baseline and x-height information. This dictionary is pre-computed based on a large sample of text-lines with baseline and x-heights


Building a Knowledge Base from Parsed Definitions
0
50
100
150
0
200
400
600
800
1000
1200

(a) Original text-line image



(b) x-height map of the text-line



(c) baseline map of the text-line

Figure B.2: Extraction of x-height and baseline of a text-line in The process of converting a text-line image into a 1D sequence. (a) shows the original text-line image. (b) shows a map of predicted locations of x-height, and (c) shows a map of predicted locations of the baseline. Note that the x-height is determined correctly for capital letters.

derived from alignment of the text-line images with textual ground-truth, together with information about the relative position of Latin characters to the baseline and x-height. Note that for some shapes (e.g., p/P, o/O), the baseline and x-height information may be ambiguous; the information is therefore stored in the form of probability densities given a connected component shape. The connected components do not need to correspond to characters; they might be ligatures or frequently touching character pairs like “oo” or “as”.

To measure the baseline and x-height of a new text-line, the connected components are extracted from the text-line and the associated probability densities for the baseline and x-height locations are retrieved. These densities are then mapped and locally averaged across the entire line, resulting in a probability map for the baseline and x-height across the entire text-line. Maps of x-height and baseline of an example text-line (Figure B.2-(a)) are shown in Figure B.2-(b) and (c) respectively. The resulting densities are then fitted with curves and are used as the baseline and x-height for line size normalization. In line size normalization (possibly curved) baseline and x-height lines are mapped to two straight lines in a fixed size output text-line image, with the pixels in between them rescaled using spline transformation. This method of normalization of a text-line has been used in the experiments for English and Fraktur.

B.4 Filter-based Normalization

The zone-based and token-dictionary methods work satisfactorily for scripts, where either baselines and x-height information is easily estimated or where segmentation can be done to extract individual characters. They fail to perform reasonably for Urdu Nastaleeq script where neither baseline nor segmentation are trivial to estimate. The filter-based normalization method is independent of estimating baseline or individual characters. This method is based on simple filter operations and affine transformation; thus making it script-independent normalization method, as compared to the normalization process described in the previous section, which was based on the shapes of the Latin alphabets. The complete normalization process is shown in Figure B.3. The input text-line image is first inverted and smoothed with a large Gaussian filter. The benefit of doing this is to capture the global structure of the underlying contents. Now, as shown in Figure B.3-(a), the smoothed image has maximum values near the center of the image along the vertical axis. These points are then fitted with a straight line (in practice, we smooth the line passing through these points as well). This is the line around which the whole text line is re-scaled using affine transformation. First a zone is found according to the difference between the height of the input image and the center line. Now, to make sure that the finally normalized image contains all the contents without clipping, the next step is to expand the image above and below of the center line by the amount equal to the height of the image. This padded image is then cropped using the zone measurement found previously. Finally, the image is scaled to the required height using affine transformation. The width of the final image is calculated by multiplying the original width with the ratio of “target” height to the height of the dewarped image. The only tunable parameter in this method is the target height. Other parameters are calculated from the given image itself. This text-line normalization is used for works reported for Urdu Nastaleeq, historical Latin and for multilingual documents. Some of the normalized images using this methodology are shown in Figure B.4.

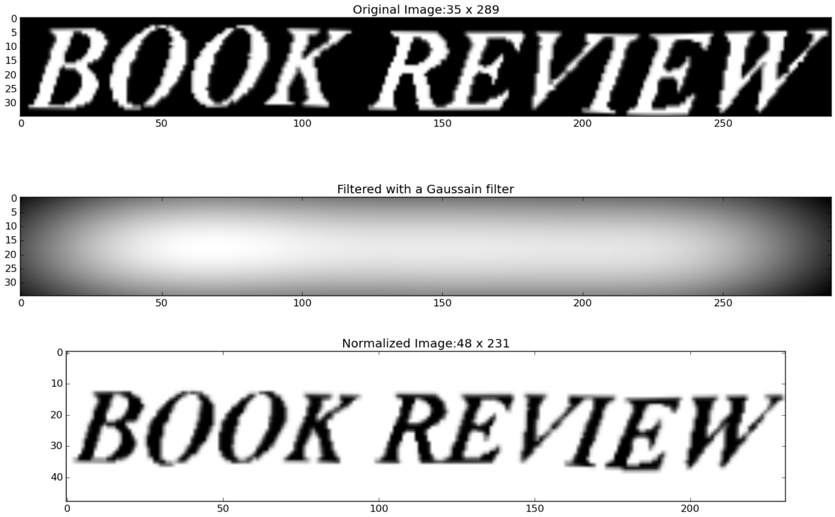

Original Image:35 x 289
BOOK REVIEW
Filtered with a Gaussain filter
Normalized Image:48 x 231
BOOK REVIEW
0
5
10
15
20
25
30
40
50
100
150
200
250

Figure B.3: Filter-based method for script-independent text-line normalization. The input text-line image is first smoothed with a large *sigma* to capture the outline of the underlying contents. This profile is then used to estimate the center line in the text-line image around which the whole text is finally rescaled. This image is rescaled using affine transformation according to the required height.

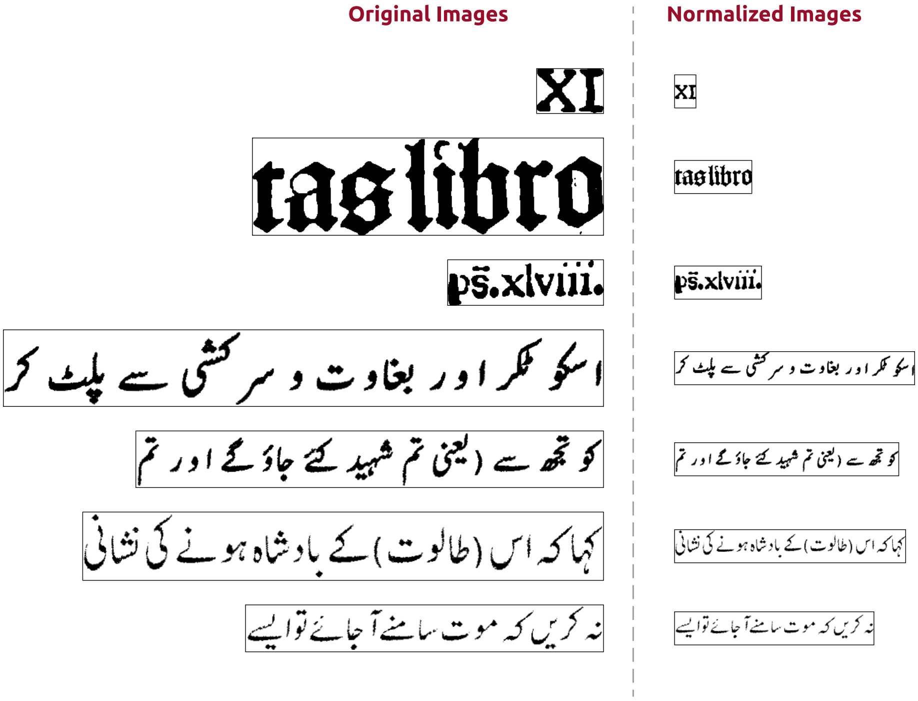

Original Images
Normalized Images
XI
taslibro
ps.xlviii.
اسکو تکبر اور بغاوت و سرکشی سے پلٹ کر
کو تجھ سے (یعنی تم شہید کیے جاؤ گے اور تم
کہا کہ اس (طالوت) کے بادشاہ ہونے کی نشانی
نہ کریں کہ موت سامنے آجائے تو ایسے

Figure B.4: SSamples text-line images normalized using filter-based method of text-line normalization. The original images (on left) are shown in their actual sizes, while those on the right are normalized to the image height of 48 pixels. The filter-based normalization performs equally well on a variety of scripts.

Bibliography

[ABB13] ABBYY, January 2013. available at http://www.abbyy.com/recognition_server.

[AHN$^+$14] Q. U. A. Akram, S. Hussain, A. Niazi, U. Anjum, and F. Irfan. Adpating Tesseract for Complex Scripts: An Example of Urdu Nastalique. In *DAS*, pages 191–195, 2014.

[Bai92] H. S. Baird. Document Image Defect Models. In H. S. Baird, H. Bunke, and K. Yamamoto, editors, *Structured Document Image Analysis*. Springer-Verlag, 1992.

[BC09] U. Bhattacharya and B.B. Chaudhuri. Handwritten numeral databases of indian scripts and multistage recognition of mixed numerals. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(3):444–457, 2009.

[BMP02] S. Belongie, J. Malik, and J. Puzicha. Shape Matching and Object Recongition using Shape Contexts. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 24(4):509–522, April 2002.

[BPSB06] U. Bhattacharya, S.K. Parui, B. Shaw, and K. Bhattacharya. Neural combination of ann and hmm for handwritten devanagari numeral recognition. In *ICFHR*, 2006.

[BR14] A. Belaid and M. I. Razzak. Middel Eastern Character Recognition. In D. Doermann and K. Tombre, editors, *Handbook of Document Image Processing and Recognition*, pages 427–457. Springer, 2014.

[BRB$^+$09] F. Boschetti, M. Romanello, A. Babeu, D. Bamman, and G. Crane. Improving OCR Accuracy for Classical Critical Editions. In *ECDL*, pages 156–167, 2009.

[Bre01] T. M. Breuel. Segmentation of Hand-printed Letter Strings using a Dynamic Programming Algorithm. In *ICDAR*, pages 821–826, sep 2001.

[Bre08] T. M. Breuel. The OCRopus Open Source OCR System. In B. A. Yanikoglu and K. Berkner, editors, *DRR-XV*, page 68150F, San Jose, USA, Jan 2008.

[BS08] T. M. Breuel and F. Shafait. AutoMLP: Simple, Effective, Fully Automated Learning Rate and Size Adjustment. In *The Learning Workshop*, January 2008.

[BSB11] S.S. Bukhari, F. Shafait, and T. M. Breuel. High Performance Layout Analysis of Arabic and Urdu Document Images. In *ICDAR*, page 1275–1279, Bejing, China, sep 2011.

[BSF94] Y. Bengio, P. Simard, and P. Frasconi. Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[BT14] H. S. Baird and K. Tombre. The Evolution of Document Image Analysis. In D. Doermann and K. Tombre, editors, *Handbook of Document Image Processing and Recognition*, pages 63–71. Springer, 2014.

[BUHAAS13] T. M. Breuel, A. Ul-Hasan, M. Al Azawi, and F. Shafait. High Performance OCR for Printed English and Fraktur using LSTM Networks. In *ICDAR*, Washington D.C. USA, aug 2013.

[CL96] R.G. Casey and E. Lecolinet. A Survey of Methods and Strategies in Character Segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(7):690–706, 1996.

[CP95] B.B. Chadhuri and S. Palit. A feature-based scheme for the machine recognition of printed Devanagari Script. 1995.

[CP97] B.B. Chaudhuri and U. Pal. An OCR system to read two Indian language scripts: Bangla and Devnagari (Hindi). In *Document Analysis and Recognition, 1997., Proceedings of the Fourth International Conference on*, volume 2, pages 1011–1015. IEEE, 1997.

[Eik93] L. Eikvil. OCR – Optical Characer Recognition. Technical report, 1993.

[EM08] M. S. M. El-Mahallway. *A Large Scale HMM-Based Omni Font-Written OCR System For Cursive Scripts*. PhD thesis, Cairo, Egypt, 2008.

[ERA57] ERA. An Electronic Reading Automation. *Electronics Engineering*, pages 189–190, 1957.

[ESB14] A. F. Echi, A. Saidani, and A. Belaid. How to separate between Machine-Printed/Handwritten and Arabic/Latin Words? *Electronic Letters on Computer Vision and Image Analysis*, 13(1):1–16, 2014.

[Fuc] M. Fuchs. The Use of Gothic OCR in processing Historical Documents. Technical report.

[Fuk80] K. Fukushima. Ncocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics*, 36:193–202, 1980.

[FV11] L. Furrer and M. Volk. Reducing OCR errors in Gothic script documents. In *Workshop on Language Technologies for Digital Humanities and Cultural Heritage*, page 97–103, Hissar, Bulgaria, September 2011.

[GAA$^+$99] N. Gorski, V. Anisimov, E. Augustin, O. Baret, D. Price, and J.-C. Simon. A2iA Check Reader: A Family of Bank Check Recognition Systems. In *ICDAR*, pages 523–526, Sep 1999.

[Gat14] B. G. Gatos. Image Techniques in Document Analysis Process. In D. Doermann and K. Tombre, editors, *Handbook of Document Image Processing and Recognition*, pages 73–131. Springer, 2014.

[GDS10] D. Ghosh, T. Dube, and A. P. Shivaprasad. Script Recognition - A Review. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32(12):2142–2161, 2010.

[GEBS04] A. Graves, D. Eck, N. Beringer, and J. Schmidhuber. Biologically Plausible Speech Recognition with LSTM Neural Nets. In Auke Jan Ijspeert, Masayuki Murata, and Naoki Wakamiya, editors, *BioADIT*, volume 3141 of *Lecture Notes in Computer Science*, page 127–136. Springer, 2004.

[GFGS06] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber. Connectionist Temporal classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *ICML*, page 369–376, 2006.

[GLF$^+$08] A. Graves, M. Liwicki, S. Fernandez, H. Bunke Bertolami, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 31(5):855–868, May 2008.

[GLS11] B. Gatos, G. Louloudis, and N. Stamatopoulos. Greek Polytonic OCR based on Efficient Character Class Numbver Reduction. In *ICDAR*, pages 1155–1159, Bejing, China, aug 2011.

[GPTF13] D. Genzel, A. C. Popat, R. Teunen, and Y. Fujii. HMM-based Script Identification for OCR. In *International Workshop on Multilingual OCR*, page 2, Washington D.C., USA., August 2013.

[Gra] A. Graves. RNNLIB: A recurrent neural network library for sequence learning problems. `http://sourceforge.net/projects/rnnl/`.

[Gra12] A. Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer, 2012.

[GS05] A. Graves and J. Schmidhuber. Framewise Phoneme Classification with Bidirectional LSTM Networks. In *IJCNN*, pages 2047–2052, Montreal, Canada, 2005.

[GS08] A. Graves and J. Schmidhuber. Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors, *NIPS*, page 545–552. Curran Associates, Inc., 2008.

[GSL$^+$] B. Gatos, N. Stamtopoulos, G. Louloudis, G. Sfikas, G. Retsinas, V. Papavassiliou, F. Simistira, and V. Katsouros. GROPLY-DB: An Old Greek Polytonic Documents Image Database. In *ICDAR*, pages 646–650, Nancy, France, August.

[Han62] W. J. Hannan. R. C. A. Multifont Reading Machine. *Optical Character Recognition*, pages 3–14, 1962.

[HBFS01] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient Flow in Recurrent Nets: The Difficulty of Learning Long-Term Dependencies. In S. C. Kremer and J. F. Kolen, editors, *A Field Guide to Dynammical Recurrent Neural Networks*. IEEE Press, 2001.

[HHK08] Md. A. Hasnat, S. M. M. Habib, and M. Khan. A High Performance Domain Specific OCR for Bangla Script. In T. Sobh, editor, *Novel Algorithms and Techniques in Telecommunication, Automation and Industrial Electronics*, page 174–178. Springer, 2008.

[HS97] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[HW65] D. H. Hubel and T. N. Wiesel. Receptive Fields and Functional Architecture in Two Nonstriate Visual Ara (18 and 19) of the Cat. *Journal of Neurophysiology*, 28:229–289, 1965.

[IH07] M. Ijaz and S. Hussain. Corpus Based Urdu Lexicon Development. In *CLT*, Peshawar, Pakistan, 2007.

[III14] IIIT, June 2014. `http://ltrc.iiit.ac.in/corpus/corpus.html`.

[IOK63] T. Iijima, Y. Okumura, and K. Kuwabara. New Process of Character Recognition using Sieving Method. *Information and Control Research*, 1(1):30–35, 1963.

[JBB$^+$95] L. Jackel, M. Battista, H. Baird, J. Ben, J. Bromley, C. Burges, E. Cosatto, J. Denker, H. Graf, H. Katseff, Y. Le-Cun, C. Nohl, E. Sackinger, J. Shamilian, T. Shoemaker, C. Stenard, I. Strom, R. Ting, T. Wood, and C. Zuraw. Neural-Net Applications in Character Recognition and Document Analysis. Technical report, 1995.

[JDM00] A.K. Jain, R. P W Duin, and Jianchang Mao. Statistical pattern recognition: a review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(1):4–37, Jan 2000.

[JKK03] C.V. Jawahar, M.N.S.S.K. Kumar, and S.S.R. Kiran. A Bilingual OCR for Hindi-Telugu Documents and its Applications. In *ICDAR*, volume 3, pages 408–412, 2003.

[KC94] G. E. Kopec and P. A. Chou. Document Image Decoding using Markov Source Models. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 16(6):602 – 617, 1994.

[KH99] T. Kanungo and R. M. Haralick. An Auotmatic Closed-Loop Methodology for Generating Character Groundtruth for Scanned Documents. *IEEE Trans. on Pattern Analysis and Machine Intellignece*, 21(2):179–183, 1999.

[KK02] D.-W. Kim and T. Kanungo. Attributed Point Matching for Automatic Groundtruth Generation. *Int. Journal on Document Analysis and Recognition*, 5(1):47–66, 2002.

[KNSG05] S. Kompalli, Sankalp Nayak, S. Setlur, and V. Govindaraju. Challenges in OCR of Devanagari Documents. In *ICDAR*, pages 327–331 Vol. 1, Seol, Korea, Aug 2005.

[KRM$^+$05] T. Kanungo, P. Resnik, S. Mao, D. W. Kim, and Q. Zheng. The Bible and Multilingual Optical Character Recognition. *Communication of the ACM*, 48(6):124–130, Jun 2005.

[KSKD15] I. U. Khattak, I. Siddiqui, S. Khalid, and C. Djeddi. Recognition of Urdu Ligatures - A Holistic Approach . In *ICADR*, Nancy, France, aug 2015.

[KUHB15] T. Karayil, A. Ul-Hasan, and T. M. Breuel. A Segmentation-Free Approach for Printed Devanagari Script Recognition. In *ICDAR*, pages 946–950, Nancy, France, aug 2015.

[LBK+98] Z. A. Lu, I. Bazzi, A. Kornai, J. Makhoul, P. S. Natarajan, and R. Schawartz. A Robust, Language-Independent OCR System. In *AIPR Workshop: Advancement in Computer-Assisted Recognition*, pages 96–104, 1998.

[LC89] Y. Le-Cun. Generalization and Network Desgin Strategies. *Connectionisms in Perspective*, Jun 1989.

[LRHP97] J. Liang, R. Rogers, R. M. Haralick, and I.T. Philips. UW-ISL Document Image Analysis Toolbox: An Experimental Environment. In *ICDAR*, page 984–988, Ulm, Germany, aug 1997.

[LSN+99] Z. Lu, R. M. Schwartz, P. Natarajan, I. Bazzi, and J. Makhoul. Advances in BBN BYBLOS OCR System. In *ICDAR*, pages 337–340, 1999.

[Lu95] Y. Lu. Machine Printed Character Segmentation — An Overview. *Pattern Recognition*, 28(1):67 – 80, 1995.

[MAM+08] S. Mozaffari, H. Abed, V. Märgner, K. Faez, and A. Amirshahi. IfN/Farsi-Database: A Database of Farsi Handwritten City Names. In *ICFHR*, page 397–402, Montreal, Canada, aug 2008.

[Mat15] J. Matas. Efficient character skew rectification in scene text images. In *ACCV*, volume 9009, page 134. Springer, 2015.

[MGS05] S. Marinai, M. Gori, and G. Soda. Artificial Neural Networks for Document Analysis and Recognition. *IEEE Trans. on Pattern Analysis and Machine Intellignece*, 27(1):23–35, January 2005.

[MSLB98] J. Makhoul, R. Schwartz, C. Lapre, and I. Bazzi. A Script-Independent Methodology for Optical Character Recognition. *Pattern Recognition*, 31(9):1285 – 1294, 1998.

[MSY92] S. Mori, C. Suen, and K. Yamamoto. Historical Review of OCR Research and Development. *IEEE*, 80(7):1029–1058, 1992.

[MWW13] Y. Mei, X. Wang, and J. Wang. An Efficient Character Segmentation Algorithm for Printed Chinese Documentation. In *UCMA*, pages 183–189, 2013.

[Nag92] G. Nagy. At the Frontiers of OCR. *IEEE*, 80(7):1093–1100, 1992.

[NHR+13] S. Naz, K. Hayat, M.I. Razzak, M.W. Anwar, S.A. Madani, and S.U. Khan. The Optical Character Recognition of Urdu-like Cursive Scripts. *Pattern Recognition*, 47(3):1229–1248, 2013.

[NHR+14] S. Naz, K. Hayat, M.I. Razzak, M.W. Anwar, S.A. Madani, and S.U. Khan. Challenges in Baseline Detection of Arabic Script Based Languages, 2014.

[NLS+01] P. Natarajan, Z. Lu, R. M. Schwartz, I. Bazzi, and J. Makhoul. Multilingual Machine Printed OCR. *International Journal on Pattern Recognition and Artificial Intelligence*, 15(1):43–63, 2001.

[NS14] N. Nobile and Y. Suen. Text Segmentation for Document Recognition. In D. Doermann and K. Tombre, editors, *Handbook of Document Image Processing and Recognition*, pages 257–290. Springer, 2014.

[NUA+15] S. Naz, A. I. Umar, R. Ahmad, S. B. Ahmed, S. H. Shirazi, and M.I. Razzak. Urdu Nastaĺiq Text Recognition System Based on Multidimensional Recurrent Neural Network and Statistical Features. *Neural Computing and Applications*, 26(8), 2015.

[OCR15] OCRopus, January 2015. available at https://github.com/tmbdev/ocropy.

[OHBA11] M. A. Obaida, M. J. Hossain, M. Begum, and M. S. Alam. Multilingual OCR (MOCR): An Approach to Classify Words to Languages. *Int'l Journal of Computer Applications*, 32(1):46–53, October 2011.

[Pal04] U. Pal. Indian Script Character Recognition: A Survey. *Pattern Recognition*, 37:1887–1899, 2004.

[PC97] U. Pal and B.B. Chaudhuri. Printed Devanagari script OCR system. *VIVEK-BOMBAY-*, 10:12–24, 1997.

[PC02] U. Pal and B. B. Chaudhuri. Identification of different script lines from multi-script documents. *Image Vision Computing*, 20(13-14):945–954, 2002.

[PD14] U. Pal and N. S. Dash. Script Identification. In D. Doermann and K. Tombre, editors, *Handbook of Document Image Processing and Recognition*, pages 291–330. Springer, 2014.

[PMM+02] M. Pechwitz, S.S. Maddouri, V. Märgner, N. Ellouze, and H. Amiri. IfN/ENIT-Database of Handwritten Arabic Words. In *CIFED*, page 129–136, Hammamet, Tunisia, oct 2002.

[Pop12] A. C. Popat. Multilingual OCR Challenges in Google Books, 2012.

[PS05] U. Pal and A. Sarkar. Recognition of Printed Urdu Text. In *ICDAR*, pages 1183–1187, 2005.

[PV10] M. C. Padma and P. A. Vijaya. Global Approach For Script Identification Using Wavelet Packet Based Features. *International Journal of Signal Processing, Image Processing And Pattern Recognition*, 3(3):29–40, 2010.

[Rab89] L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[Ras10] Rashid, S.F. and Shafait, F. and Breuel, T. M. Discriminative Learning For Script Recognition. In *ICIP*, Hong Kong, sep 2010.

[Ras14] S. F. Rashid. *Optical Character Recognition – A Combined ANN/HMM Approach*. PhD thesis, Kaiserslautern, Germany, 2014.

[RDL13] R. Rani, R. Dhir, and G. S. Lehal. Script Identification of Pre-segmented Multi-font Characters and Digits. In *ICDAR*, Washington D.C., USA, August 2013.

[RH89] W.S. Rosenbaum and J.J. Hilliard. System and Method for Deferred Processing of OCR Scanned Mail, 07 1989.

[SAL09] R. Smith, D. Antonova, and D. S. Lee. Adapting the Tesseract Open Source OCR Engine for Multilingual OCR. In *Int. Workshop on Multilingual OCR*, July 2009.

[SCPB13] N. Sharma, S. Chanda, U. Pal, and M. Blumenstein. Word-wise script identification from video frames. In *ICDAR*, pages 867–871, Washington D.C. USA, Aug 2013.

[Sen94] A. W. Senior. *Offline Cursive Handwriting Recognition using Recurrent Neural Networks*. PhD thesis, England, 1994.

[SHNS09] M. Sagheer, C. He, N. Nobile, and C. Suen. A New Large Urdu Database for Off-Line Handwriting Recognition. page 538–546, Vietri sul Mare, Italy, sep 2009.

[SIK$^+$09] F. Slimane, R. Ingold, S. Kanoun, A. M. Alimi, and J. Hennebert. A New Arabic Printed Text Image Database and Evaluation Protocols. In *ICDAR*, page 946–950, Barcelona, Spain, July 2009.

[SJ12] N. Sankaran and C.V. Jawahar. Recognition of printed Devanagari text using BLSTM Neural Network. In *ICPR*, pages 322–325, nov 2012.

[SJ15] A.K. Singh and C.V. Jawahar. Can RNNs Reliably Separate Script and Language at Word and Line Level? In *ICDAR*, pages 976–980, Nancy, France, August 2015.

[SKB08] F. Shafait, D. Keysers, and T. M. Breuel. Efficient Implementation of Local Adaptive Thresholding Techniques Using Integral Images. In B. A. Yanikoglu and K. Berkner, editors, *DRR-XV*, page 681510, San Jose, USA, Jan 2008.

[SLMZ96] R. Schwartz, C. LaPre, J. Makhoul, and Y. Zhao. Language-Independent OCR Using a Continuous Speech Recognition System. In *ICPR*, page 99–103, Vienna, aug 1996.

[SM73] R. Sinha and K. Mahesh. A Syntactic Pattern Analysis System and Its Application to Devanagari Script Recognition. 1973.

[Smi07] R. Smith. An Overview of the Tesseract OCR Engine. In *ICDAR*, pages 629–633, 2007.

[Smi13] R. Smith. History of the Tesseract OCR Engine: What Worked and What Didnt In *DRR-XX*, San Franciso, USA, Feb 2013.

[SP00] J. Sauvola and M. Pietikäinen. Adpative Document Image Binarization. *Pattern Recognition*, 33:225–236, 2000.

[Spi97] A. L. Spitz. Multilingual Document Recognition. In H. Bunke and P. S. P. Wang, editors, *Handbook of character Recognition and Document Image Analysis*, pages 259–284. World Scientific Publishing Company, 1997.

[SPS08] B. Shaw, K.S. Parui, and . Shridhar. Offline Handwritten Devanagari Word Recognition: A holistic approach based on directional chain code feature and HMM. In *ICIT*, pages 203–208. IEEE, 2008.

[SS13] N. Sabbour and F. Shafait. A Segmentation-Free Approach to Arabic and Urdu OCR. In *DRR-XX*, San Francisco, CA, USA, feb 2013.

[SSR10] T. Saba, G. Sulong, and A. Rehman. A Survey on Methods and Strategies on Touched Characters Segmentation . *International Journal of Research and Reviews in Computer Science*, 1(2):103–114, 2010.

[SUHKB06] F. Shafait, A. Ul-Hasan, D. Keysers, and T.M. Breuel. Layout analysis of urdu document images. In *Multitopic Conference, 2006. INMIC '06. IEEE*, pages 293–298, Dec 2006.

[SUHP+15] F. Simistira, A. Ul-Hasan, V. Papavassiliou, B. Gatos, V. Katsouros, and M. Liwicki. Recognition of Historical Greek Polytonic Scripts Using LSTM Networks. In *ICDAR*, page 766–770, Nancy, France, aug 2015.

[Sut12] I. Sutskever. *Training Recurrent Neural Networks*. PhD thesis, Dept. of Computer Science, Univ. of Toronto, 2012.

[SYVY10] R. Singh, C.S. Yadav, P. Verma, and V. Yadav. Optical Character Recognition (OCR) for Printed Devnagari Script Using Artificial Neural Network. *International Journal of Computer Science & Communication*, 1(1):91–95, 2010.

[Tau29] G. Tauscheck. Reading machine, 12 1929.

[Tes14] Tesseract, June 2014. `http://code.google.com/p/tesseract-ocr/`.

[TNBC00] K. Taghva, T. Nartker, J. Borsack, and A. Condit. UNLV-ISRI document collection for research in OCR and information retrieval. In *DRR–VII*, page 157–164, San Jose CA, USA, 2000.

[UHAS+15] A. Ul-Hasan, M. Z. Afzal, F. Shafait, M. Liwicki, and T. M. Breuel. A Sequence Learning Approach for Mutliple Script Identification. In *ICDAR*, pages 1046–1050, Nancy, France, 2015.

[UHASB13] A. Ul-Hasan, S. B. Ahmed, F. Shafait, and T. M. Breuel. Offline Printed Urdu Nastaleeq Script Recognition with Bidirectional LSTM Networks. In *ICDAR*, pages 1061–1065, Washington D.C., USA., Aug 2013.

[UHB13] A. Ul-Hasan and T. M. Breuel. Can we Build Language Independent OCR using LSTM Networks? In *International Workshop on Multilingual OCR*, page 9, Washington D.C., USA., Aug 2013.

[USHA14] S. Urooj, S. Shams, S. Hussain, and F. Adeeba. CLE Urdu Digest Corpus. In *CLT*, Karachi, Pakistan, 2014.

[vBSB08] J. van Beusekom, F. Shafait, and T. M. Breuel. Automated OCR Ground Truth Generation. In *DAS*, page 111–117, Nara, Japan, sep 2008.

[VGSP08] G. Vamvakas, B. Gatos, N. Stamatopoulos, and S. Perantonis. A Complete Optical Character Recognition Methodology for Historical Documents. In *DAS*, page 525–532, Nara, Japan, sep 2008.

[Vit67] A.J. Viterbi. Error bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269, April 1967.

[Wer90] P. Werbos. Backpropagation Through Time: What Does It Do and How to Do It. In *Proceedings of IEEE*, volume 78, page 1550–1560, 1990.

[Whi13] N. White. Training Tesseract for Ancient Greek OCR. *Eutypon*, pages 1–11, 2013.

[WLS10] N. Wang, L. Lam, and C. Y. Suen. Noise Tolerant Script Identification of Printed Oriental and English Documents Using a Downgraded Pixel Density Feature. In *ICPR*, pages 2037–2040, 2010.

[YSBS15] M. R Yousefi, M. R. Soheili, T. M. Breuel, and D. Stricker. A Comparison of 1D and 2D LSTM Architectures for Recognition of Handwritten Arabic (Accepted for publication). In *DRR–XXI*, San Francisco, USA, 2015.

[YY94] S.J. Young and S. Young. The HTK Hidden Markov Model Toolkit: Design and Philosophy. *Entropic Cambridge Research Laboratory, Ltd.*, 2:2–44, 1994.

Bibliography

Adnan Ul-Hasan

Work Experience

Current	Senior Engineer at Pakistan Atomic Energy Commission, *Pakistan*
January 2014 January 2016	PhD Student at DFKI Kaiserslautern, Germany Competence Center, *Multimedia Analysis and Data Mining (MADM)*
April 2011 December 2013	PhD Student at TU Kaiserslautern, Germany *Image Understanding & Pattern Recognition (IUPR)* Research Group
November 2011 March 2013	Research Assistant at Perception Labs., *TU Kaiserslautern* Development of stimuli for the experiments related to human perception
December 2006 December 2010	Senior Engineer at Pakistan Atomic Energy Commission, *Pakistan*
September 2004 November 2006	Juior Engineer at Pakistan Atomic Energy Commission, *Pakistan*

Education

January 2016	PhD in Computer Science, **TU Kaiserslautern**, Germany Thesis: "Generic Text Recognition using Long Short-Term Memory Neural Networks" Advisor: Prof. Dr. Andreas Dengel
September 2004	Masters of Science in Systems Engineering 3.21/4.0, **Pakistan Institute of Engineering & Applied Sciences (PIEAS)**, Islamabad, Pakistan Thesis: "Investigation of Cardiovascular Functioning using the Non-Linear Characteristics of Human ECG" Advisor: Prof. Dr. Syed Mohammad Arif
September 2002	Bachelor of Science in Electrical Engineering 79.7% (with honors), **University of Engineering and Technology (UET)**, Taxila, Pakistan

Scholarships and Certificates

March 2014	Faculty Scholarship for post-graduate studies in Germany (€15000)
March 2010	DAAD Scholarship for post-graduate studies in Germany (€48000)
Aug. 2002	PAEC Scholarship for graduate studies in Pakistan (Pak. Rs. 200000)

Computer Skills

Languages: Python, MATLAB, Octave
Office: LaTeX, MS and Open Office, MS Project, MS Visio
OS: Ubuntu, Windows

Publications

1. **Adnan Ul-Hasan**, Syed Saqib Bukhari, Andreas Dengel, *"Meaningless Text OCR Model for Medieval Scripts"*, 2nd International Conference on Natural Sciences and Technology in Manuscript Analysis, 2016, Hamburg, Germany.
2. **Adnan Ul-Hasan**, Syed Saqib Bukhari, Andreas Dengel, *"OCRoRACT: A Sequence Learning OCR System Trained on Isolated Characters"*, 12th International Workshop on Document Analysis Systems, 2016, Santorini, Greece. (Oral presentation)
3. Fallak Asad, **Adnan Ul-Hasan**, Faisal Shafait, *"High Performance OCR for Camera-Captured Blurred Documents with LSTM Networks"*, 12th International Workshop on Document Analysis Systems, 2016, Santorini, Greece. (Oral presentation)
4. **Adnan Ul-Hasan**, Faisal Shafait, Marcus Liwicki, *"Curriculum Learning for Printed Text Line Recognition of Ligature-based Scripts"*, 13th International Conference on Document Analysis and Recognition, 2015, Nancy, France.
5. **Adnan Ul-Hasan**, Muhammad Zeshan Afzal, Faisal Shafait, Marcus Liwicki, Thomas M. Breuel, *"A Sequence Learning Approach for Multiple Script Identification"*, 13th International Conference on Document Analysis and Recognition, 2015, Nancy, France.
6. Tushar Karayil, **Adnan Ul-Hasan**, Thomas M. Breuel, *"A Segmentation-Free Approach for Printed Devanagari Script Recognition"*, 13th International Conference on Document Analysis and Recognition, 2015, Nancy, France. (Oral presentation)
7. Fotini Simistira, **Adnan Ul-Hasan**, Vassilis Papavassiliou, Basilis Gatos, Vassilis Katsouros, Marcus Liwicki, *"Recognition of Historical Greek Polytonic Scripts Using LSTM Networks"*, 13th International Conference on Document Analysis and Recognition, 2015, Nancy, France. (Oral presentation)
8. Mayce Ali Al-Azawi, **Adnan Ul-Hasan**, Marcus Liwicki, Thomas M. Breuel, *"Character-Level Alignment Using WFST and LSTM for Post-processing in Multi-script Recognition Systems - A Comparative Study"*, 11th International Conference on Image Analysis and Recognition, Oct. 2014, Portugal.
9. **Adnan Ul-Hasan**, Thomas M. Breuel, *"Can We Build Language Independent OCR using LSTM Networks"*, 4th International Workshop on Multilingual OCR, Aug. 2013, Washington D.C., USA. (Oral presentation)
10. **Adnan Ul-Hasan**, Saad bin Ahmed, Faisal Rashid, Faisal Shafait, Thomas M. Breuel, *"Offline Printed Urdu Nastaleeq Script Recognition with Bidirectional LSTM Networks"*, 12th International Conference on Document Analysis and Recognition, ICDAR-2013, Aug., 2013, Washington D.C., USA. (Oral Presentation)
11. Thomas M. Breuel, **Adnan Ul-Hasan**, Mayce Ali Al-Azawi, Faisal Shafait, *"High Performance OCR for Printed Latin and Fraktur using LSTM Networks"*, 12th International Conference on Document Analysis and Recognition, ICDAR-2013, Aug., 2013, Washington D.C., USA.
12. **Adnan Ul-Hasan**, Syed Saqib Bukhari, Sheikh Faisal Rashid, Faisal Shafait, Thomas M. Breuel, *"Semi-Automated OCR Database Generation for Complex Scripts"*, 21st International Conference on Pattern Recognition, Nov., 2012, Tsukuba, Japan. (Oral presentation)
13. **Adnan Ul-Hasan**, Syed Saqib Bukhari, Faisal Shafait, Thomas M. Breuel, *"OCR-Free Table of Contents Detection in Urdu Books"*, 10th IAPR International Workshop on Document Analysis Systems, DAS-2012, March, 2012, Gold Coast, Queensland, Australia.
14. Faisal Shafait, **Adnan Ul-Hasan**, Daniel Keysers, Thomas M. Breuel, *"Layout Analysis of Urdu Document Images"*, 10th IEEE International Multi-topic Conference, Dec. 2016, Islamabad, Pakistan.
15. Faisal Shafait, Muhammad Usman, **Adnan Ul-Hasan**, Habibullah Jamal, Shoab A. Khan, *"An Architecture of 2-D IDCT for Real Time Decoding of MPEG/JPEG Compliant Bit-Streams"*, 17th IEEE International Conference on Micro-Electronics, Dec. 2005, Islamabad, Pakistan.