# Statistical Language Modeling for Historical Documents using Weighted Finite-State Transducers and Long Short-Term Memory

Dissertation

submitted to the

Department of Computer Science at Technical University of Kaiserslautern

for the fulfillment of the requirements for the doctoral degree

Doctor of Natural Sciences

(Dr. rer. nat.)

by

Mayce Al Azawi

Thesis supervisors:
Prof. Dr. Thomas M Breuel, Google Inc.
Prof. Dr. Andreas Dengel, DFKI Kaiserslautern
apl. Prof. Dr. Marcus Liwicki, DFKI Kaiserslautern

Supervisory committee:
Prof. Dr. Markus Nebel, TU Kaiserslautern (Chair)
Prof. Dr. Karsten Berns, TU Kaiserslautern

Kaiserslautern, 02 February, 2015

**D 386**

TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

# Abstract

The goal of this work is to develop statistical natural language models and processing techniques based on Recurrent Neural Networks (RNN), especially the recently introduced Long Short-Term Memory (LSTM). Due to their adapting and predicting abilities, these methods are more robust, and easier to train than traditional methods, i.e., words list and rule-based models. They improve the output of recognition systems and make them more accessible to users for browsing and reading. These techniques are required, especially for historical books which might take years of effort and huge costs to manually transcribe them.

The contributions of this thesis are several new methods which have high-performance computing and accuracy. First, an error model for improving recognition results is designed. As a second contribution, a hyphenation model for difficult transcription for alignment purposes is suggested. Third, a dehyphenation model is used to classify the hyphens in noisy transcription. The fourth contribution is using LSTM networks for normalizing historical orthography. A size normalization alignment is implemented to equal the size of strings, before the training phase. Using the LSTM networks as a language model to improve the recognition results is the fifth contribution. Finally, the sixth contribution is a combination of Weighted Finite-State Transducers (WFSTs), and LSTM applied on multiple recognition systems. These contributions will be elaborated in more detail.

Context-dependent confusion rules is a new technique to build an error model for Optical Character Recognition (OCR) corrections. The rules are extracted from the OCR confusions which appear in the recognition outputs and are translated into edit operations, e.g., insertions, deletions, and substitutions using the Levenshtein edit distance algorithm. The edit operations are extracted in a form of rules with respect to the context of the incorrect string to build an error model using WFSTs. The context-dependent rules assist the language model to find the best candidate corrections. They avoid the calculations that occur in searching the language model and they also make the language model able to correct incorrect words by using context-dependent confusion rules. The context-dependent error model is applied on the university of Washington (UWIII) dataset and the Nastaleeq script in Urdu dataset. It improves the OCR results from an error rate of 1.14% to an error rate of 0.68%. It performs better than the state-of-the-art single rule-based which returns an error rate of 1.0%.

This thesis describes a new, simple, fast, and accurate system for generating correspondences between real scanned historical books and their transcriptions. The alignment has many challenges, first, the transcriptions might have different modifications, and layout variations than the original book. Second, the recognition of the historical books have misrecognition, and segmentation errors, which make the alignment more difficult especially the line breaks, and pages will not have the same correspondences. Adapted WFSTs are designed to represent the transcription. The WFSTs process Fraktur ligatures and adapt the transcription with a hyphenations model that allows the alignment with respect to the varieties of the hyphenated words in the line breaks of the OCR documents. In this work, several approaches are implemented to be used for the alignment such as: text-segments, page-wise, and book-wise approaches. The approaches are evaluated on German calligraphic (Fraktur) script historical documents dataset from "Wanderungen durch die Mark Brandenburg" volumes (1862-1889). The text-segmentation approach returns an error rate of 2.33% without using a hyphenation model and an error rate of 2.0% using a hyphenation model. Dehyphenation methods are presented to remove the hyphen from the transcription. They provide the transcription in a readable and reflowable format to be used for alignment purposes. We consider the task as classification problem and classify the hyphens from the given patterns as hyphens for line breaks, combined words, or noise. The methods are applied on clean and noisy transcription for different languages. The Decision Trees classifier returns better performance on UWIII dataset and returns an accuracy of 98%. It returns 97% on Fraktur script.

A new method for normalizing historical OCRed text using LSTM is implemented for different texts, ranging from Early New High German 14th - 16th centuries to modern forms in New High German applied on the Luther bible. It performed better than the rule-based word-list approaches. It provides a transcription for various purposes such as part-of-speech tagging and n-grams. Also two new techniques are presented for aligning the OCR results and normalize the size by using adding Character-Epsilons or Appending-Epsilons. They allow deletion and insertion in the appropriate position in the string. In normalizing historical wordforms to modern wordforms, the accuracy of LSTM on seen data is around 94%, while the state-of-the-art combined rule-based method returns 93%. On unseen data, LSTM returns 88% and the combined rule-based method returns 76%. In normalizing modern wordforms to historical wordforms, the LSTM delivers the best performance and returns 93.4% on seen data and 89.17% on unknown data.

In this thesis, a deep investigation has been done on constructing high-performance language modeling for improving the recognition systems. A new method to construct a language model using LSTM is designed to correct OCR results. The method is applied on UWIII and Urdu script. The LSTM approach outperforms the state-of-the-art, especially for unseen tokens during training. On the UWIII dataset, the LSTM returns reduction in OCR error rates from 1.14% to 0.48%. On the Nastaleeq script in Urdu dataset, the LSTM reduces the error rate from 6.9% to 1.58%.

Finally, the integration of multiple recognition outputs can give higher performance than a single recognition system. Therefore, a new method for combining the results of OCR systems is explored using WFSTs and LSTM. It uses multiple OCR outputs and votes for the best output to improve the OCR results. It performs better than the ISRI tool, Pairwise of Multiple Sequence and it helps to improve the OCR results. The purpose is to provide correct transcription so that it can be used for digitizing books, linguistics purposes, N-grams, and part-of-speech tagging. The method consists of two alignment steps. First, two recognition systems are aligned using WFSTs. The transducers are designed to be more flexible and compatible with the different symbols in line and page breaks to avoid the segmentation and misrecognition errors. The LSTM model then is used to vote the best candidate correction of the two systems and improve the incorrect tokens which are produced during the first alignment. The approaches are evaluated on OCRs output from the English UWIII and historical German Fraktur dataset which are obtained from state-of-the-art OCR systems. The Experiments show that the error rate of ISRI-Voting is 1.45%, the error rate of the Pairwise of Multiple Sequence is 1.32%, the error rate of the Line-to-Page alignment is 1.26% and the error rate of the LSTM approach has the best performance with 0.40%.

The purpose of this thesis is to contribute methods providing correct transcriptions corresponding to the original book. This is considered to be the first step towards an accurate and more effective use of the documents in digital libraries.

# Acknowledgment

With a great pleasure, I would like to thank my doctoral advisor Prof. Thomas Breuel for guiding me throughout my Ph.D. work.

I would like to thank him for introducing me to the OCR, language modeling, and historical documents field, for giving me the chance to do research, and work in different projects in his group, teaching, and also giving the honor to manage the TextGrid project. Thanks a lot for his support, encouragement to do a creative work in my Ph.D thesis to compete big companies.

I would like to thank Prof. Andreas Dengel for giving me a great chance to work in his group, his advice, and support.

I would like to thank Prof. Marcus Liwicki for his advice during this thesis. I would like to thank him for his support, encouragement, and motivation. Their comments were always very useful in improving this work. I would like to thank Prof. Hans Hagen and Dr. Bernd Schürmann for their support during my time at TU-Kaiserslautern. I would like to thanks Prof. Markus Nebel and Prof. Karsten Berns for agreeing to be in my defense commission.

Thanks to Mrs. Ingrid Romani and Mrs. Gabriele Sakdapolrak for their always-ready-to-help attitude and their efforts in helping me with administration work.

I would like to thank my parents for their love, support, and keep believing in me. I would like to thank them for encouragement, their patience throughout my Ph.D. work, and for providing me with a good education that led to this dissertation. I would like to thank my brothers and my sister for their love and support. Many thanks for my best friends Margret Lorig and Sarah Al-benna.

I would also like to thank all the colleagues in the IUPR lab, MADM, KM group at DFKI for stimulating discussions and friendly time. Thanks a lot for all people working in the department, adminstration, RHRK, Mensa, and libraries at TU-Kaiserslautern.

Mayce

# Erklärung

Sehr geehrte Damen und Herrn,
Hiermit erkläre ich, dass ich die Dissertation mit dem Titel:

“Statistical Language Modeling for Historical Documents using Weighted Finite-State Transducers and Long Short-Term Memory”

selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel genutzt habe. Alle wörtlich oder inhaltlich übernommenen Stellen habe ich als solche gekennzeichnet.
Ich versichere außerdem, dass ich die beigefügte Dissertation in keinem anderen Promotionsverfahren eingereicht habe.

Mit freundlichen Grüßen
Mayce Al Azawi

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation

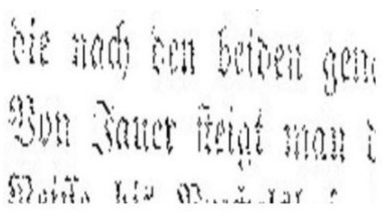
An important goal of historical document analysis are digital libraries for historical critical editions. Therefore, information extraction, natural language processing, and corpus analysis techniques should be performed. One prerequisite is OCR which transcribes the input document image to a machine readable format. A satisfactory OCR accuracy for the whole content of digital documents will allow us to lower the costs for manually transcribing the documents. Various historical digitalization projects have used OCR systems, e.g., the Improving Access to Text (IMPACT) project[1] used the ABBYY system[2], the TextGrid project[3] has integrated the OCRopus system[4] to obtain recognition results. This dissertation is devoted to the improvements of OCR results of such systems.

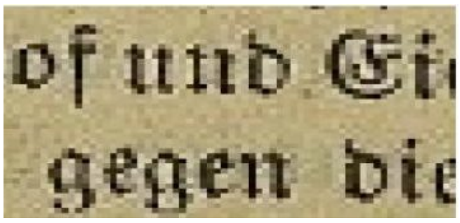
In recognition systems such as OCR and Handwritten Recognition systems, errors appear in the results due to noisy and complex documents, bad quality, and low resolution [Lop09]. Document layout analysis is the process of identifying and categorizing the regions of interest in the scanned image of a text document. A reading system requires the segmentation of text zones from non-textual ones and the arrangement in their correct reading order. Over- and under-segmentation of lines and characters are produced due to the large variety of layouts and narrow spaces. The challenges of document layout analysis are an open problem for historical and handwritten documents. Much research has been done, but the recognition task is still difficult. The documents are bad quality images due to blurred, yellow papers, faded ink, damaged parchment, or ink bleed-through [Fis12, AD07], as shown in Figure 1.1.

Moreover, historical and handwritten documents have connected characters which create the so-called ligatures such as the German calligraphic Fraktur script as in Figure 1.2(a), 1.2(b),

[1] IMPACT Project: `http://www.impact-project.eu/taa/tech/tools/`
[2] ABBYY: `http://www.abbyy.de/`
[3] TextGrid Project: `http://textgrid.de/`
[4] OCRopus Project: `https://github.com/tmbdev/ocropy`

(a) Unknown font to the classifiers


(b) Bad binarization


(c) Low resolution

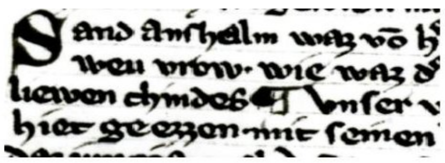

(d) Historical orthography

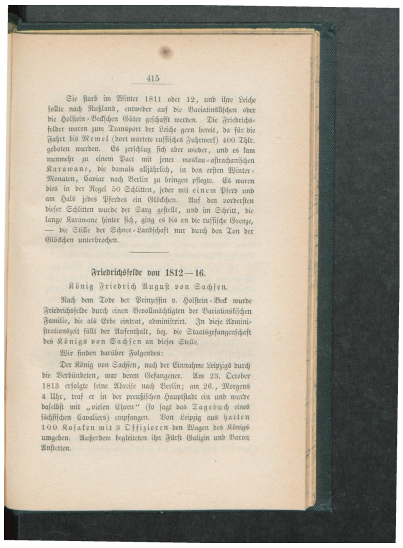
Figure 1.1: Problems of historical documents.

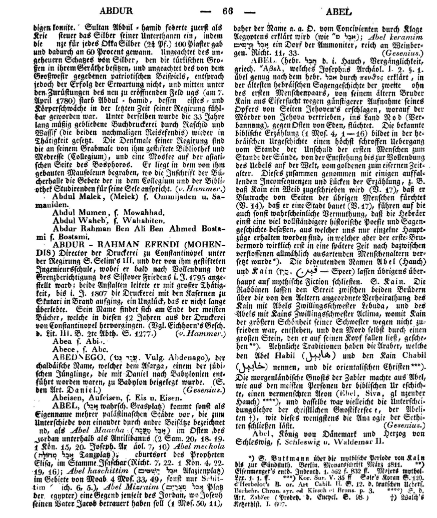
1.2(d), and 1.2(e). In the historical Fraktur script, some Glyphs are extended vertically and rise above the x-height, which refers to the distance between the baseline and the mean line in a typeface. Typically, this is the height of the letter $x$ in the font, as well as the $u, v, w,$, and $z$. For example, the Letter $f$ has a ascender height and connects with the letters which have descender heights. The small letters $p, h, g, y$, and Fraktur long $s$ exceeds the descender height and extends to the next text line. The capital letters $A, P, Q, Y$ also have the same descender expansion.

The connected characters or merged text lines are difficult to split for any automatic segment system. The character segmentation is important for classifiers which learn the character segments, for example, Multilayer Perceptrons (MLP) [Ros62]. The Hidden Markov Model (HMM) and Neural Networks (NN) based recognizers are segmentation-free, they learn sequences that do not depend on character segmentation prior to recognition. They are widely used in speech and handwriting recognition [Rab89, GLF$^+$09].

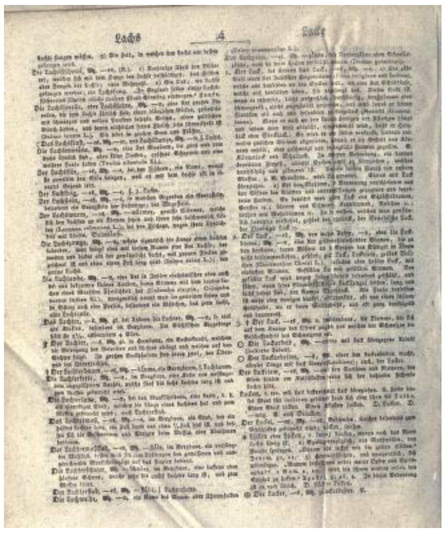
As for historical documents, large electronic text corpora with a consistent orthography are often unavailable for historical languages [AAAB13]. Furthermore, recognition models cannot easily be transferred to different manuscripts in case of non-standardized orthography, and special character symbols (fonts unknown to a trained system). Historical documents usually include multiple scripts in one volume. For example, the "Allgemeine Encyclopdie der Wissenschaften und Künste" published by Johann Samuel Ersch and Johann Gottfried Gruber, also known as the "Ersch-Gruber". Ersch-Gruber documents have German Fraktur, Greek, Latin, Arabic, and Hebrew scripts, as shown in Figure 1.2(b).

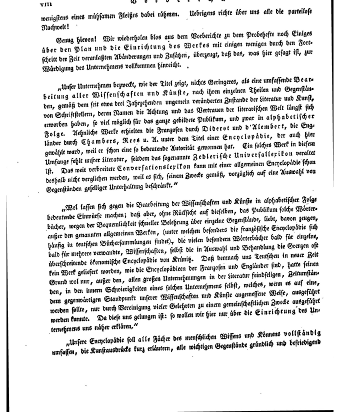
This dissertation solves the problem of language modeling by using various methods and

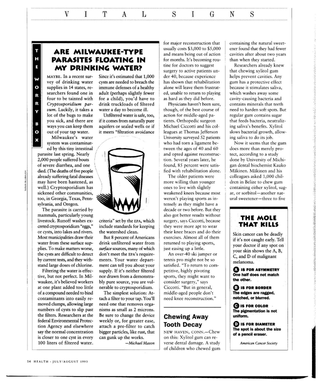

(a) Fontane's document


(b) Ersch-Gruber

(c) Luther Bible


(d) Campe dictionary


(e) Fraktur document


(f) UNLV dataset

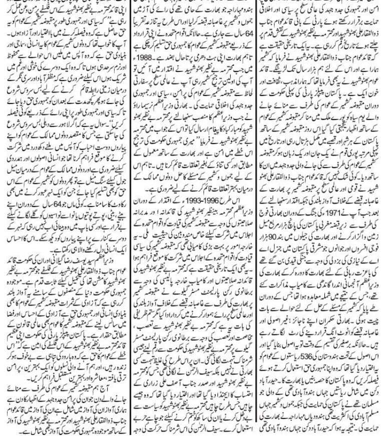

(g) Urdu dataset

Figure 1.2: Collection of document images.

alignment techniques. The remainder of this chapter is organized as follows. The challenges of historical documents are shown in Section 1.2. The hypothesis of the dissertation is stated in Section 1.3. The contributions of the dissertation are presented in Section 1.4. Finally, the outlines of the dissertation are given in Section 1.5.

## 1.2 Challenges of Historical Documents

Pre-existing OCR systems are unable to solve all those problems during their pipeline. Therefore, post-processing techniques are required after the recognition stage. In the last decade, much research has been done on error corrections in automatic speech recognition, optical character recognition, and spell-checking. A language model can be used to recognize valid word forms of a language. A language model is a model of dictionary words and, possibly, allowed word sequences. Generally, a language model is capable of creating infinite dictionary corrections for the errors. To find the correct form of word, the language model needs to search through whole entries for the best candidate corrections. This search, however, is time consuming and inefficient. Furthermore, it is not eligible for unknown word forms in the dictionary.

In order to suggest appropriate candidate corrections, an error model is required to generate the correct form of the misspelled or misrecognized words. Example of recognition results of Urdu and Fraktur script are given in Figure 1.3(a). The Figure shows the misrecognized words and the appropriate corrections. The traditional error model is the Levenshtein-Damerau edit distance algorithm attributed initially by [Lev66b]. In the edit distance algorithm, the misrecognition is assumed to be a finite number of operations applied to characters of a string: deletion, insertion, and substitution. Schulz et al. [SM02] introduced a deterministic Levenshtein automata for correcting misspelled words using an electronic dictionary. The error model using finite-state automaton and is widely used in speech recognition [Moh03a], in spell-checker [PH12, HNH08], and in OCR [LNCPCA10].

A second major problem in historical documents analysis is linking the transcripts of historical documents to the corresponding positions in the original document [KMA04, FFFB11, FM06, FIFB11], as shown in Figure 1.3(b). The alignment produces a huge amount of training data with aligned image segments and transcriptions which are necessary in order to create good recognition models. Fischer et al. [FIFB11] used alignment based on HMM. For transcription alignment, it is necessary to prepare, and clean-up the transcriptions before use in alignment, as shown in Figure 1.3(c). Therefore, dehyphenation methods are implemented to support clean-up of transcription. They can be used for different purposes, such as building n-grams and text corpora. The work of [GT94] eliminates the hyphen from the text, ignores the combined words, and Unicode strings.

A third problem is the alignment of different word forms. With non-standard and unknown orthography in historical documents, words-list, and rule-based approaches are designed to normalize the historical word forms to modern word forms, and vice versa [BPD11a]. These approaches are unable to cover all the variety of orthographical and misrecognized tokens. Therefore a new generation of learning-based language modeling is implemented to solve these problems [AAAB13]. Figure 1.3(d) shows the original document image, historical form with the corresponding modern form.

Fourth problem is the combination of multiple recognition systems as several different recognizers exist is implemented to improve the recognition results [LR11, LZ97, BRB+09], as shown in Figure 1.3(e). These methods are based on heuristic algorithms to find the best results from the n-best paths and are unable to predict the correct candidates which are unavailable in the OCR systems. They vote for an incorrect character, if all the systems provided incorrect characters in the same position of the sequences.

The purpose is to provide correct digitization and correspondence transcriptions. This is the first step towards accurate and more effective use of the documents in digital libraries. Therefore, this dissertation introduces seven novel contributions, as described in Section 1.4.

## 1.3 Hypothesis

The main target of the dissertation is to improve and normalize the recognition results. The hypotheses are:

H1 WFSTs are suitable for constructing error models to build orthographical or OCR correction rules.

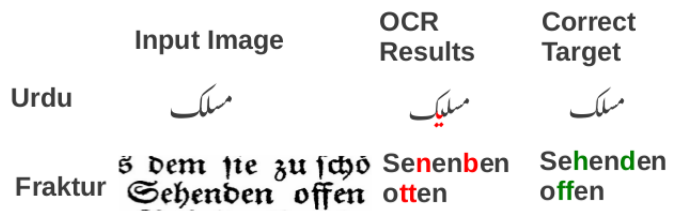
H2 WFSTs are suited for alignment between the scanned historical documents and the transcription. They adapt the transcription according to the OCR errors and make it more flexible on character, word, line, and page level so that it can be compatible with various books.

H3 Noisy transcriptions can be cleaned and converted to hyphenation free text by using machine learning methods. These methods are capable of classifying the hyphen as a line break or in combined words or noise.

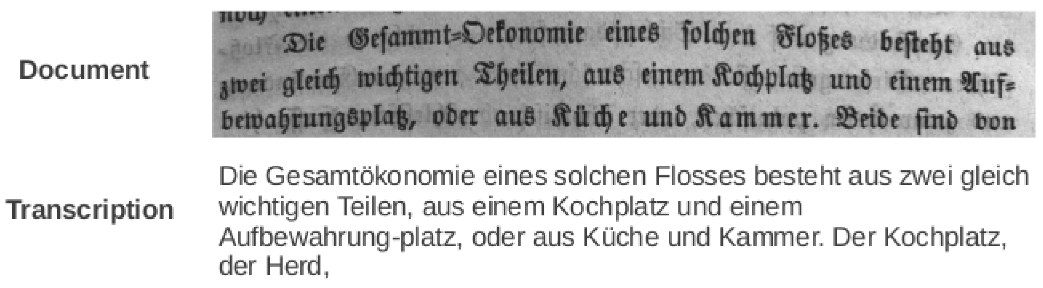
H4 Language modeling can be done with NN for modern and historical documents. Such learning-based language modeling can predict the historical orthographical rules which are inconsistent in historical text.

H5 Neural Networks can be used to improve the OCR accuracy by correcting the recognition results.

H6 WFSTs are a flexible alignment for combining multiple OCR outputs and LSTM are able to predict the errors appearing in several OCR systems.



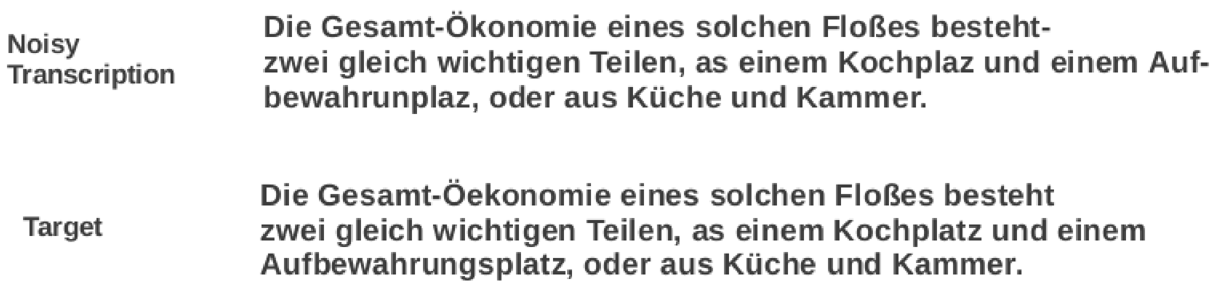
(a) Correction of recognition output


Document
Transcription

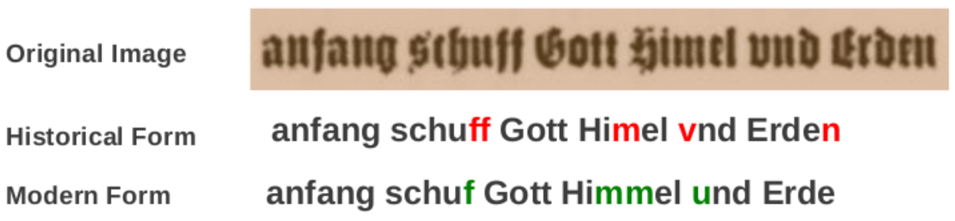
Die Geſammt=Oekonomie eines ſolchen Floßes beſteht aus zwei gleich wichtigen Theilen, aus einem Kochplatz und einem Auf= bewahrungsplatz, oder aus Küche und Kammer. Beide ſind von

Die Gesamtökonomie eines solchen Flosses besteht aus zwei gleich wichtigen Teilen, aus einem Kochplatz und einem Aufbewahrung-platz, oder aus Küche und Kammer. Der Kochplatz, der Herd,

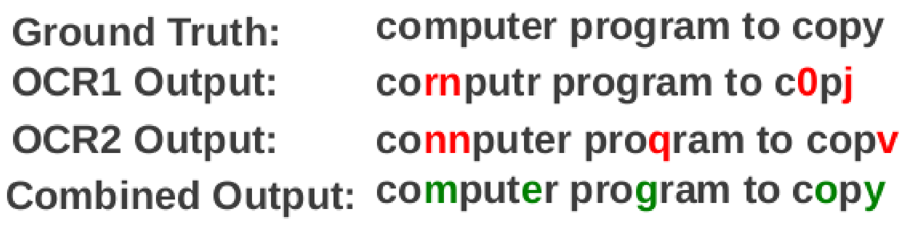
(b) Alignment of books to transcriptions


Noisy Transcription
Die Gesamt-Ökonomie eines solchen Floßes besteht-
zwei gleich wichtigen Teilen, as einem Kochplaz und einem Auf-
bewahrunplaz, oder aus Küche und Kammer.
Target
Die Gesamt-Öekonomie eines solchen Floßes besteht
zwei gleich wichtigen Teilen, as einem Kochplatz und einem
Aufbewahrungsplatz, oder aus Küche und Kammer.

(c) Dehyphenation and clean-up


Original Image
anfang ſchuff Gott Himel vnd Erden
Historical Form
anfang schuff Gott Himel vnd Erden
Modern Form
anfang schuf Gott Himmel und Erde

(d) Normalize historical orthography


Ground Truth: computer program to copy
OCR1 Output: cornputr program to c0pj
OCR2 Output: connputer proqram to copv
Combined Output: computer program to copy

(e) Combination of multiple recognition output

Figure 1.3: Problems and targets

## 1.4 Contributions of this Dissertation

The purpose is to develop statistical language modeling based on WFST and LSTM for modern and historical scripts written in English, German, and Urdu. Below is a summary of the work carried out.

C1 Error Model using context-dependent rules (Chapter 3). A novel new technique to correct the OCR errors by means of WFST with context-dependent confusion rules. It is implemented to address H1.

- The OCR confusions which appear in the recognition outputs are translated into edit operations, e.g., insertions, deletions, and substitutions using Levenshtein edit distance algorithm to construct an error model.
- The error model avoids the calculations that occur in searching the language model.
- It makes the language model eligible to correct incorrect words.
- It is language independent. It is applied on English and Urdu.
- It has no limited words size.
- It deals with various numbers and types of errors.
- It is faster needs less memory space and returns a lower error rate than the state-of-the-art single rule-based approach [HNH08].

C2 Transcription Alignment (Chapter 4)

- The approach aligns the manuscript of historical books and the transcription that have no matching information about the correspondence at the word, line, and sentence level. It is implemented to address H2.
- It contributes a new hyphenation model for hyphenating the transcriptions according to the scanned books.
- The alignment model works under OCR errors. OCR errors make the textual contents of the books differ from the original or corresponding transcriptions.
- It aligns with various transcriptional modifications.
- It avoids the problems of recognition, modified transcriptions, and segmentation errors by using edit operations on character level (insertion, deletion, substitution).
- Those edit operations are translated to edit rules adapt the manuscript and the transcription during the alignment.
- It works under segmentation errors and different line breaks.
- WFSTs process Fraktur ligatures.

- It provides a graphical user interface to view the alignment pairs, make them accessible, and easy to edit.

C3 Dehyphenation by Classification (Chapter 4). It is used for noisy historical and modern transcriptions and addresses in H3.

- The approach is applied on English script from UWIII dataset and Fraktur script from documents of Theodor Fontane.
- Dehyphenation removes the hyphen and converts the results of Handwriting Recognition (HWR) or OCR into a readable and reflowable formats.
- It processes Unicode, e.g, words with Umlaut. It also processes hyphenated pairs and combined words which were not solved by [GT94].
- It is robust against OCRs errors which include recognition errors and substitution of symbols, i.e., dashes with hyphen due the misrecognition.

C4 A novel learning-based for normalizing historical orthography (Chapter 5) is introduced which addresses in H4.

- It achieves better results than the state-of-the-art methods [BPD11a] even with the absence of consistent orthographic conventions in historical text.
- There is no fixed lexicon accessed by orthographic form is available.
- It works well with seen and unseen tokens, especially the state-of-the-art methods that cannot normalize the unseen tokens where the spelling of words often differ in various way.
- It is evaluated on OCRed scripts from $14^{th}$ to $16^{th}$ centuries.
- Simple approach requires less time and effort to train.

C5 Language-Independent (Chapter 6) is a new techniques. It is introduced to correct the OCR errors by means of Recurrent Neural Network (RNN) with LSTM, as addresses in H5.

- It is applied on challenging script Urdu.
- It is applied on OCR results of the Latin script and Urdu UPTI datasets.
- New LSTM model avoids the calculations that occur in searching the language model.
- It makes the language model eligible to correct seen and unseen incorrect words.
- Prediction using LSTM is fast and accurate. Other approaches are time consuming and labor-intensive [HNH08].

C6 A Heuristic-based alignment approach for combining multiple recognition systems is introduced (Chapter 7) as addresses in H6.

- Flexible combination and adaptation using WFST.
- The purpose is combining the output of many OCRs to improve the recognition results.
- WFST approach aligns multiple OCR systems and avoids the problem of segmentation and recognition errors.
- It performs better than the state of the art ISRI [RJN96] and Wemhoener et al. [WYM13].
- It is language independent approaches.

C7 A novel learning-based model for combining multiple recognition systems using LSTM (Chapter 7) is implemented, as addresses in H6.

- Combining the output of many OCRs to improve the recognition results.
- The approach is flexible and adaptable.
- The combination of OCRs outputs is improved using LSTM.
- It is a language independent approach.
- It performs better than the state-of-the-art [RJN96, WYM13].
- The heuristic approaches are unable to predict characters which did not appear in the OCR results. They also are unable to correct a character, if all the OCR systems provide the misrecognized version of this character. LSTM approach solves all those problems.

## 1.5 Dissertation Overview

Figure 1.4 shows the structure of this dissertation. Chapter 2 presents a survey about the state-of-the-art in statistical language modeling which is used in OCR, Automatic Speech Recognition (ASR), and Machine Translation (MT). It also described the open-source toolkits.

Next, Part I presents language modeling methods using WFSTs. Then, Chapter 3 introduces an error model to support the statistical language modeling using WFSTs. Chapter 4 provides a method for aligning historical books with their transcriptions and hyphenation model using WFSTs. This chapter presents a dehyphenation models by classification for historical noisy transcription.

Part II shows the abilities of the LSTM which outperforms other language modeling methods. Chapter 5 contributes on normalizing historical OCR text from historical document of the 14th century from Luther Bible. The goal is to build a language model maps the historical word forms to modern word forms. Chapter 6 provides extensive empirical comparison of results obtained with various advanced language modeling techniques using rules-based and

LSTM. The experiments are applied on Urdu and UWIII datasets obtained from OCR system. Chapter 7 focuses on the combination of WFSTs and LSTM to combine the results of various OCR systems. Chapter 8 summarizes the achieved results and concludes the work.

Finally, the appendix introduces the WFSTs and their related algorithms as well as RNN and the architecture of the LSTM.

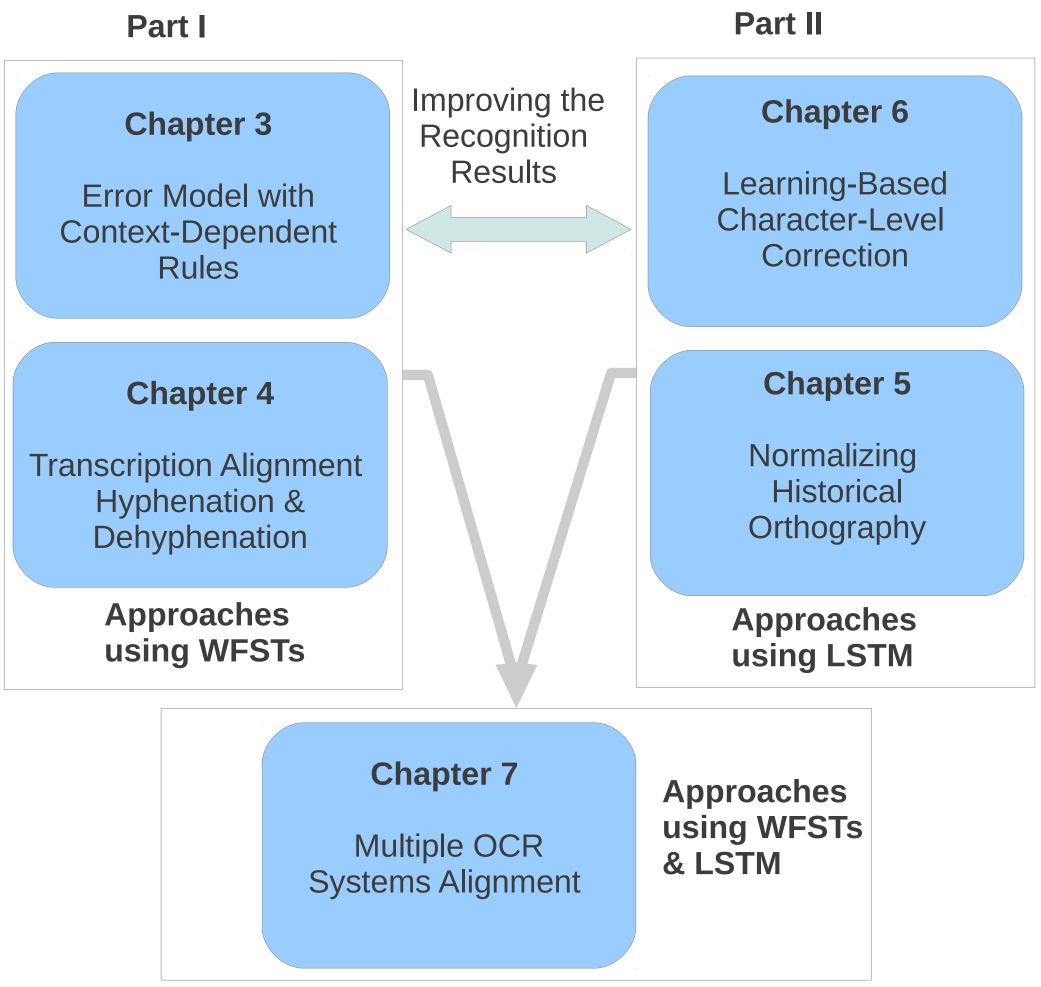

Part I
Part II
Chapter 3
Error Model with Context-Dependent Rules
Improving the Recognition Results
Chapter 6
Learning-Based Character-Level Correction
Chapter 4
Transcription Alignment Hyphenation & Dehyphenation
Chapter 5
Normalizing Historical Orthography
Approaches using WFSTs
Approaches using LSTM
Chapter 7
Multiple OCR Systems Alignment
Approaches using WFSTs & LSTM

Figure 1.4: Visualization of the structure of this dissertation illustrating the relationship between different chapters and their contribution to different areas of OCR post-processing. Filled blocks show the tools and the areas to which this dissertation contributes.

# Chapter 2

# State-of-the-Art Language Modeling

## 2.1 Motivation

In this Chapter, an overview of the language modeling (LM) approaches and tasks will be given. The Chapter will summarize selected recognition systems, such as OCR, Automatic Speech Recognition (ASR), and Machine Translation (MT).

The strategies and methods of building LMs are described. Also the toolkits, and open-source libraries will be shown. Beside corrections task, there are several tasks which need LMs to perform the alignment with documents for various purposes, such as normalizing historical orthography, generate image to transcription segments, and combining recognition results among many systems. These task will be described with their strategies.

The Chapter is structured as follows: Section 2.2 discusses the LMs which are designed to automatic speech recognition, and machine translation. In Section 2.3, the LMs in top OCR systems are explained. Section 2.4 shows several LMs which are used for solving problems in OCR. New strategies for building learning-based LM are explained in Section 2.4.2. The open-source tools are shown in Section 2.5. Summary and discussion will be in Section 2.6.

## 2.2 Language Modeling in Automatic Speech Recognition (ASR) and Machine Translation (MT)

In speech recognition, the work of Mohri et al. [Moh03a] presented a general algorithm based on classical, and new weighted automata algorithms for exactly computing the edit distance between two string distributions given by two weighted automata. In their work, a new and general synchronization algorithm is published for weighted transducers. It combined with Epsilon-removal and can be used to normalize weighted transducers with bounded delays. The work also made use of composition, Epsilon-removal of weighted transducers, and the determinization of weighted automata. The work of Allauzen et al. [AM09] proposed linear-space algorithms for computing the edit distance between a string, and an arbitrary weighted automaton, or an unambiguous weighted automaton. The algorithm is efficient and optimized for finding an optimal alignment of a string, and such as a weighted automaton. It helped to understand the classical algorithms [MCL07, Gus97], and making possible to generalize them [Moh03a].

The open-source Weighted Finite-State library is developed and achieved a competitive performance for building and applying the WFST for the LM. The work of [CAM07] presented general algorithms for building and optimizing transducer models. They explained the application of these methods to large-vocabulary recognition tasks and their experimental results. In Machine translation, [KB03] presented a derivation of the alignment template model for statistical machine translation and an implementation of the model using WFSTs.

Mikolov et. al [MKB$^+$11] extended the RNN LM for speech recognition task and compared to feedforward networks. The backpropagation through time algorithm (BPTT) is used for learning. In their work, they showed the number of trained classes over the epochs and time. The RNN with Kneser Ney (KN) reduce the perplexity to 107 using 8000 classes, in 107 Min/epoch, and for testing 148 Sec/test. The perplexity is a measurement of how well a probability model predicts a sample. It is used to evaluate LMs. The network has 200 hidden units. While the perplexity is increased from 109 to 134 using 6000 to 30 classes, the training and testing time was decreased. The RNN results was better using interpolated 5-grams model with KN smoothing, and no count cutoffs. Four RNN networks were trained with 250, 300, 350, and 400 units in the hidden layer. During the training phase, the complexity is increased with the amount of the steps, and stayed constant during the testing. The simple RNN outperforms the standard feedforward network, while the BPTT provide a better improvement. In their work, they have implemented a simple factorization of the output layer using classes, to avoid the computational bottleneck between the hidden and output layers, and to reduce the size of the weight matrix.

## 2.3 Language Modeling in Top OCRs

In this section, the LM approaches of the top recognition systems are described. Those approaches are investigated in different research groups and their results were reported in Breuel et al. [BUHAAS13] [1]. A report of the recent novel methods will be also described.

### 2.3.1 OCRopus

OCRopus is a free document analysis and optical character recognition (OCR) system released under the Apache License, Version 2.0 with a very modular design through the use of plugins in 2007. These plugins allow OCRopus to swap out components easily. OCRopus is developed under the lead of Thomas Breuel at TU-Kaiserslautern, Germany and is sponsored by Google.

[1] Co-authored by the author of this thesis

OCRopus is an OCR system that combines pluggable layout analysis, pluggable character recognition, and pluggable LM. It aims primarily for high-volume document conversion, namely for Google Book Search, but also for desktop, and office use [Bre08].

Breuel et al. [Bre08] has shown in OCRopus 0.4 the RAST-Based layout analysis, MLP recognizer, and the recognition outputs are presented as multiple paths WFST for each text line. The input label is the character or ligature segment and the output label is the recognition output. LMs based on WFSTs can be composed modularly from dictionaries, n-grams, grammatical patterns, and semantic patterns. This allows OCRopus to be re-targeted and adapted quickly to new document types and languages. Statistical LMs associate probabilities with strings; their function in an OCR system is to resolve ambiguous or missing characters to their most likely interpretation.

The early work of Breuel [T B95] showed that LM can be phrase-based or word-based. The phrase-based models are suitable when the data set consists of short phrases. A dictionary of phrases can be learned from a large corpus and the frequencies of the phrases can be used to approximate probability of the word $P(W)$. But the phrase-based model is not scalable and is not suitable for most applications. Word-based models are effected by errors in segmentation and by the presence of characters like "/", ".", etc. Breuel et al. [T B94] made a comparison between phrase and word based models. It was found that phrase-based models performed poorly when many of the phrases in the test set were not part of the LM. As compared to the phrase model, the performance of the unigram word model was poor due to the large entropy, and perplexity of the word model. The performance was seen to improve when the output of recognizers based on the two LMs was combined. For handwritten recognition system, two statistical LMs were constructed. The first consists of phrases and the estimated associated frequency information. The second consists of unconstrained concatenations frequently used words, separated by spaces, and uses a word insertion penalty to assign probabilities to phrases [13]. It has higher coverage 87% than the phrase-based LM but contains many implausible or impossible phrases. The objective of recognition in a Bayesian framework is to find $P(W)$ can be obtained from the LM. The work of Breuel et al. [T B95] manipulated the output of the segmentation stage into hypothesis graph to allows skips, insertions, and the deletions. Finding the best path through the hypothesis graph constrained by the dictionary is carried out using the Viterbi algorithm [Vit67].

In OCRopus 0.7, Breuel et al. [BUHAAS13] implemented, and integrated LSTM recognizer for text line recognition. The LSTM yields better results compared to the state-of-the-art OCR systems. An OCR-Service was provided to the partners of the TextGrid project using OCRopus. The OCR-Service recognized historical documents with Fraktur script.

### 2.3.2 Tesseract

Tesseract is an optical character recognition engine for various operating systems. It is free software, released under the Apache License, Version 2.0, and development by Ray Smith has been sponsored by Google since 2006.

The dictionaries in Tesseract are represented by Directed Acyclic Word Graph (DAWG). The dictionaries include the pre-generated system dictionary, the document dictionary, and the a user-provided words list. The implementation of the data structure has been improved to parallelize the search over all the DAWGs and support multi-language text [Smi07]. The most frequent punctuations and numbers are encoded in DAWGs. For unambiguous dictionary words, the shape classifier must identify the a clear winner among all the alternative choices for the word. The recognition result is considered and stop further processing for the word, if there is no dictionary word word exists. By having a text file including a word per line, the DAWD will be created during the creation of the train data for the language [Smi11]. Smith et al. [Smi11] has discussed that the LM with frequency-based dictionary could be more damage than helpful. The paper showed an analysis of the contrariety with the help of the Google Books n-grams corpus and concludes that noisy-channel models that closely model the underlying classifier and segmentation errors are required [Smi11]. They proposed an isolated shape classifier, combined with a LM that has a word n-gram frequency model ($1 <= n <= 3$) or a binary n-gram dictionary model.

The paper showed the limitation of the frequency models because the close relationship between the probability that a word occurs in the language with given some context for n-gram within $n > 1$ and the probability that a word is correct. The study included weak and accurate classifiers. The most frequent word is best choice, if the classifier is weak. And it is better to involve the previous $n - 1$ words. If the classifier is accurate, then the most probable word is not the best choice, even in context with previous words.

They also reported that the LM is less effective for Latin language such as English, because the OCR shape classifier is more accurate. While LM is required for high error rate systems such as ASR, OCR for Arabic, and Hindi.

Smith et al. [LS12] has described a combination of two parallel correction paths using document-specific image and LMs.

The models are adapted to shape and vocabularies within a book to build the correction hypotheses. The models is depending on selecting the correct words.

The purpose of the shape model is to solve the confusion in similar shapes such as $m/rn$ and $1/I/l$ and also assist the LM to distinguish between between *back* and *blue* while correcting the words. The paper showed the usage of the strength of each model to solve confusions.

The tokens are passed over all the instances, then the more likely answer are taken. However, this could generate valid, and false hypotheses of out of vocabulary tokens in ambiguous context. Therefore, each instance of correction is evaluated by the image model to remove mismatches.

The tokens are verified after correction, the LM is adapted with a new word-list, and the image model updates the shape clusters. Improvement can obtained over iterations and during their experiments then found that good performance shown first iteration.

The approach is evaluated on scanned books. In training, 2 million words are used, 6 million words are used in the testing. The system was able to reduce the word error rate of Tesseract by 25% on a large test set.

### 2.3.3 ABBYY FineReader

ABBYY FineReader is a commercial OCR software is intended to simplify converting paper documents to digital data. It recognizes printed text in 190 languages in modern and historical form[2].

ABBYY OCR system was required for recognizing historical texts printed with Fraktur script for the period 1800 1938 in Meta-E project. The Meta-E project focused on providing technology basis for digitisation and web-publishing of valuable printed sources spanning several centuries of European history. The linguistic part of the project was done by ATAPY Software[3], ABBYY's long-term partner in OCR, and linguistic development. ATAPY joined the Meta-E project to build Language Models for 5 old European languages: Old English, Old French, Old German, Old Italia, and Old Spanish.

The recognition system of ABBYY OCR works by analysing a text image and making a hypothesis. The hypothesis is about which letter or word an image represents. Then, the hypotheses are analysed in context, and verified by using sophisticated OCR dictionaries made up of LMs.

ABBYY OCR system uses the latest findings in linguistic technologies and morphological dictionaries in the background so that the system can take a decision for ambiguous recognized characters, example a misrecognized character "ü", the OCR system is assisted by linguistic information, addition information for identifying the language is German, and dictionaries to predict the correct text. This technique works well in moder text, but not easy to apply in the historical text, and depends on the available dictionaries from that time.

ABBYY FineReader checks all words' spelling in order to highlight the incorrectly recognized characters and build correct spelling hypotheses for operator to choose from. The dictionaries are not only sets of words. The dictionaries are built in optimized way which each word is stored as a single database entry in the form of a linguistic formula which an appropriate grammar paradigm is assigned. Based on authentic dictionaries, and original old European texts, a study is done on the historical material, building the word stock, and assigning appropriate paradigm formulas of words.

To analyse the dictionaries and the original text of a give period, they selected 10 dictionaries

[2] ABBYY: `http://www.abbyy.de/`
[3] ATAPY: `http://www.atapy.com/`

representing the state of project languages in the range of the publication's years from 1808 to 1930. The analysis is done using 105 authentic books belonging to that period, of more than 50 MB of archived information in volume. The historical LMs were built on the basis of the coexisting FineReader models and reduced the big efforts of building a new ones. However, the stage of building historical LMs appeared laborious, as ATAPY engineers had to manually compare the information from the authentic dictionaries, and texts to existing FineReader models which are about 500.000 entries. 159 historic grammar paradigms were added to carry out corrections that were missing in the contemporary models. The LMs were tested on the original old text material collected during the analysis stage. The results are 98,91% vocabulary coverage for Old English, 99,16% for Old French, 96,58% for Old German, 98,58% for Old Italian, and 98,79% for Old Spanish language. The new OCR functionality is available for purchase as an additional module to ABBYY FineReader 7.0.

### 2.3.4 Raytheon BBN Technology

The research group of Bolt, Beranek, and Newman (BNN) described the enhancement to their HMM-based OCR system that results in a significant increase in the speed of the system without any impact on recognition accuracy. The LM which is used was an n-gram on character level. In [LBK+99], BBN presented a language-independent OCR system performing open-vocabulary OCR on Arabic, English, and Chinese. The system is based on Hidden Markov Models (HHMs) and utilizes the same advanced technology that is used for speech recognition. They described that the HMM-based OCR system was easily trainable on new sets of data and portable to recognize new scripts. They showed that the system depends on the estimation of character models, as well as a lexicon, and grammar, from training data. The character models are estimated by taking the feature vectors and the corresponding ground truth. Then, each text line is transcribed in terms of words, therefore the character model component also can make use of a lexicon obtained from a large text corpus. Their grammar LM is estimated from the same text or a larger set of text. The training process used the orthographic rules that depend on the type of script. For example, the rules tell whether the text lines go horizontally or vertically, as in traditional Chinese, and if vertically, whether the text is read from left-to-right, as in Roman script, or right-to-left, as in Arabic script.

They compute a character HMM, a lexicon, and a LM. These models are estimated from large text corpora which are independent of the image data. The LM is usually a bigram or trigram model that contains the probabilities of all pairs or triples of words. The recognition process is mainly a search for the most likely sequence of characters given the input feature vector sequence, the lexicon, and the LM. Since the Viterbi algorithm would be quite expensive when the state space includes a large vocabulary and a bigram or trigram LM, they used a multi-pass search algorithm. They used character trigram and word unigram LMs for Arabic script. Without a lexicon, a Character Error Rate (CER) of 4.7% was obtained. With a 30K-lexicon, a CER of 3.3% was obtained using our unlimited-vocabulary technique which solved the out-of-vocabulary (OOV) problem, a 30% relative improvement. They used the DARPA Arabic OCR Corpus with 89-character set to train and test their omnifont system.

## 2.4 Recent Developments in Language Modeling for OCR

There are several tasks of post-processing for documents and recognition systems which need a LM for correcting, normalizing, and aligning purposes.

### 2.4.1 Automatic Document and Transcription Alignment

Earlier work by Manmatha [KMA04] proposes an alignment using HMMs or dynamic time warping (DTW). Text alignment with historical documents is concerned by the problem of alignment when line by line correspondences are available and page by page correspondences are provided. This approach semi-automatically aligns word-images with ASCII words without OCR. The approach is that in some situations the stretch between two anchor words may be relatively large, making the dynamic programming somewhat expensive. Another HMM-based algorithm for transcription alignment to automatically segmented handwritten manuscript was proposed in Fischer et al. [FFFB11]. It was designed for word by word alignment of scanned handwritten document images with ASCII transcripts. However, this model was not hierarchical, and not practical for aligning large sequences.

A hierarchical HMM-based automatic evaluation of OCR accuracy for a digital library of books [FM06] is recent work from two alignment approaches using HMM. This approach is similar to the previous HMM model. Unlike Krogh's linear HMMs, the HMM at each level of their hierarchical alignment approach directly takes positions as states, and calculates the probability of generating a sequence of OCR output given any possible sequence of positions in the ground truth. There is a state correspondence to every position in the ground truth sequence. In this approach, they sought to align text to text, and not text to images. Given the problem and domain differences, the transition probabilities and generative probabilities have to be, and are defined differently. Compared with edit distance based alignment algorithm, the HMM based alignment allows one to learn the domain knowledge through training over aligned or even unaligned sequences, and formulate the probabilities of alignments using arbitrary distributions. It is more flexible and powerful.

The authors of Fischer et al. [FIFB11] propose a multi-pass alignment method based on HMMs that combines text line recognition, bigram LM, string alignment, and keyword spotting to cope with word substitutions, deletions, and insertions in the transcription. In a segmentation-free approach, transcriptions of complete pages are aligned with sequences of text line images. They applied their approach on the Parzival data set. However, this approach does not take text line segmentation errors into account, and it assumes page breaks to be present in the transcription. This method was applied at page-level and is segmentation-free in the sense that neither segmentation of text line images into words nor line break information in the transcription are needed. A possible speed-up of text alignment was proposed in Yalniz et al. [YM11]. This approach aligns the ground truth and the OCR output using a fast recursive text alignment scheme (RETAS). First, it aligns unique words from vocabulary with unique words in the OCR output. This process is recursively applied to each text segment between the matching unique words until the text segments become small. Then the edit distance based alignment is used to align those (short) chunks of text and obtain the final alignment. This approach segments the alignment problem into small subproblems, which leads to significant time saving, and less computation and memory space requirements than in previous approaches. The worst case running time was achieved when there are no common unique words between OCR output and the ground truth, which means there are no anchors to make the alignment. In this case the texts have to be aligned using an exact alignment algorithm at the leaf level.

### 2.4.2 Learning-Based Language Modeling

A novel designed learning-based approach for learning recognition errors and improve the recognition results is implemented by Al Azawi et al. [AAUHLB14] using LSTM. For the LSTM based approach, powerful LSTM networks were trained to learn corrections by themselves. LSTM is specifically designed to overcome limitations of RNN. LSTM has the ability to remember the target association between irrelevant input and target events even for very long time lags. A new Character-Level alignment was proposed to normalize the strings length before the training of LSTM, as will be described in Chapter 6.

Statistical LMs based on n-grams are widely used in several application such as ASR, MT, and OCR and considered as state-of-the-art. However, n-grams are not giving results good enough even after a lot of efforts [TMC11]. Al Azawi et al. [AAAB13] implemented new LMs for historical spelling variants that are based on LSTM. The models achieved better performance in normalizing of historical text. The experiments show the new implemented LSTM model has better performance than the existing methods on the unknown data which were unseen during the training, as will be described in Chapter 5.

Frinken et al. [FZMEB$^+$12] presented to model the target language using RNN with LSTM cells. The network contains two hidden layers, a project layer, and recurrent LSTM layer. The network is trained LSTM on word features. They trained the on a given shortlist which contains the most frequent words, for each word three additional features are added: one related to out-of-shortlist (OOS) word, one for the start-of-sentence (SOS) tag, and one for the end-of-sentence (EOS) tag. The approach needed long time to process because long computations of the huge size of the feature vector. Therefore, the project layer is used to reduce the calculations due to the huge dimensionality of the word encoding. The output layer has the same dimension as in the input layer. The probabilities are estimated from the training corpus and are smoothed with bi-gram

LM. In their paper, they demonstrated that the context information of the words decreases the perplexity of a testing set using LSTM LM. This LM can use for re-ranking generated N-best lists. The experiments are conducted on the IAM off-line handwritten datasets and LOB corpus. In Zamora-Martinez et al. [ZFB$^+$14], they trained the neural network for LM task using the backpropagation. They also trained on words features as in [ZFB$^+$14]. The size of the input and output is corresponding to the vocabulary size, which the training of the neural network expensive. The size is reduced by taking a subset of the vocabulary which is called shortlist with the more frequent words. The proposed NN LM were 4-gram, 3-gram, bigram, and unigram. They used 10 K input/output shortlist of words. The NN LM is a combined of four linear combined NN. Two recognition results was used which are obtained from HMM/ANN and BLSTM to be used for the experiments. Using 55 K vocabulary, the 4-gram NN LM improved by a 18%. While it improved up to 20% by using 103 K vocabulary. The increase of the dictionary size leads to a better performance, the use of 103 K dictionary returned the best performance of NN LM. The perplexity is decreased by increasing the modeling length and using the Witten Bell smoothing [ZFB$^+$14]. The best perplexity was obtained from 4-gram LM smoothed by Witten Bell. However, that was not the case using Kneser-Ney [KN95]. The projection layer size varying in each neural network with 160, 192, 224, 256 neurons, while hidden layer size is 200 neurons.

### 2.4.3 Historical Orthography

Historical documents have nonstandard orthography. Therefore, a normalization approach is required for mapping historical form to modern form, makes the documents more accessible, and available for further processing e.g., POS taggers and simple wordform queries by other users working with the data. The goal of this research is to provide an automatic mapping from wordforms from ENHG ($14^{th}$ - $16^{th}$ centuries) to the corresponding modern wordforms from New High German (NHG), as shown in Chapter 1. Bollman et al. [BPD11b, BPD11c] compared different approaches for normalizing of variant word forms on their modern spelling using string distance measures and evaluate them on two types of historical text Luther Bible and Anselm. These approaches are either rule-based or wordlist approaches. The wordlist approach is bilingual wordlists. The wordlist maps each historical wordform to the modern wordform that it is most often aligned with. The normalized wordforms are generated using this method by substituting an old wordform with its modern counterpart as specified in the wordlist. If the word is not in the list, it is left unchanged. But this approach fails to produce a modern wordform, if it's historical wordform is not in the list.

The rule-based approach is derived by computing Levenshtein edit distance. It outputs a numerical value with respect to edit operations such as substitution, insertion, and deletion of single characters that map strings from the historical to modern wordform. If multiple rules are applicable at the same position within a word, the rule that is ranked higher is applied. This approach works well. However, the same experiments showed lower performance on unknown wordforms [BPD11b], specially in some periods (time in history) the rules were varying. The rule-based approach was widely used in Information Retrieval applied for normalizing historical language data [EGF06, AWH07, Pil05, HM11].

Jurish et al. [Jur10] evaluated historical German data for various canonicalization techniques. The techniques were based on: string identity, transliteration, phonetization, and rewrite transduction. The experiments show the best performance which is selected by a Hidden Markov Model. Zobel et al. [ZD95] proposed the usage of string matching techniques, such as similarity measures and phonetic coding, in conjunction with lexicon indexes to find approximate matches in a large lexicon. The experiments showed that n-gram string distance is a suitable fine search mechanism, because it is effective and simple to compute. VARD2 is a statistical tool for normalising spelling [BR08]. It is adapted by [HM11] to tackle the problem of spelling variations in a corpus of personal Portugese letters from the 16th to the 20th century.

Based on the modern lexicon and the 140 manually collected patterns, the set of all potential rule based historical variants can be computed automatically (hypothetical lexicon)[4]. The experiment shows the possibility of automatic mapping from rule based historical variants to their equivalent in the modern vocabulary. However, they showed, by far, not all historical variants can be described by simple replacement rules.

### 2.4.4 Error Modeling

In the last decade, a number of research have been done on error corrections in ASR, OCR, and spell-checking. A LM in simple form of one-tape finite-state automaton has been proposed to recognize a valid wordforms of a language. However, the LM should be capable of creating infinite dictionary corrections for the errors in efficient manner. To find the correct form of word, the LM needs to search through the whole entries to find the best candidate corrections which is time consuming, and inefficient. It is also not eligible for unknown wordforms in the dictionary.

In order to suggest appropriate candidate corrections, an error model is required to generate correct form of the misspelled or misrecognized words. The traditional error model is the Levenshtein-Damerau edit distance algorithm attributed initially by [Lev66b]. In the edit distance algorithm, the misrecognition is assumed to be a finite number of operations applied to characters of a string: deletion, insertion, and substitution. The Levenshtein distance [Lev66b] between two strings is used to control the search in the lexicon by generating the exact lexical words where the Levenshtein distance between the misspelling words and lexical words does not exceed a given bound [SM02]. This makes the search in lexicon more efficient and faster than previous methods.

The work of Schulz et al. [SM02] introduced two related methods for correcting misspelling

[4]http://impactocr.wordpress.com/2011/10/25/language-parallel/

words using an electronic dictionary, that is implemented as a deterministic finite-state automaton. The correction procedures are similar to Oflazers approach [Ofl96], but completely avoid the computation of Levenshtein distances. In the first method [SM02], they computed an appropriate deterministic Levenshtein automata in linear time to the length of the input. In their second method, they avoided the actual computation of a deterministic Levenshtein automaton for the input word by using precompiled tables to simulate transitions in the automaton. The precompiled table contains a parametric, generic description of states, and transitions of the automaton. The traversal of the dictionary automaton is controlled using the table improved the correction method. As an extension to the work of Schulz et al. [SM02], they suggested to use a specific costs of the edit operations based on the symbol of the operation. For example, the frequency of occurrences of a given correction candidate can be taken into account for ranking correction candidates.

Error model is widely used in different application such as spell-checkers and post-processing in speech recognition. The authors in [PH12, PSL12] proposed an error model to aid the LM in a spell-checker to correct misspelling errors and provide the user with several suggestions. The error model consists of two-tape finite-state automaton mapping any string of the error model alphabet to at least one string of the LM alphabet. Their goal is to build Finnish spell-checking. They used finite-state implementation of a traditional edit distance algorithm. Extensive research has been conducted on spell-checker [Kuk92, SM02].

There are several approaches that have been proposed to solve this problem that use statistical machine learning [GR98], lexical semantics [HB05], and web crawls [RHSM07].

Hassan et al. [HNH08] proposed an error model with edit operations and additional transposition operation, i.e., exchanging the order of two consecutive symbols "ab" $\rightarrow$ "ba". Their work was proposed for spell-checking and corrected words with only error of edit distance 1 or 2. In OCR, the work of [LNCPCA10] proposed a method to post processing the results of the OCR. They proposed a LM with error mode. The error model include the insertion, deletion, and substitution and apply it on handwritten Spanish surnames in scanned forms.

Further LM research is done in this thesis by constructing a new error model with context dependent is designed by Al Azawi et al. [AAB14a]. A simple way to use an arbitrary string-to-string relation transducers such as a misrecognized and the suggested corrections is presented. We test it by implementing a Finite-State Transducer form of the Levenshtein edit distance relation. The context-dependent confusion rules is used to build the error model. It is based on the confusion matrix that the OCR produces and is dependent on the context of the strings (OCR results). The context help to fit the confusion rule in the proper string where it belongs and brings the string to the corrections. The LM can be as simple as a list of finite words compiled into a finite-state transducers. The error model provides a efficient and fast search to assist the LM. The frequencies of a token or rule in the corpus are converted to weights in the finite-state transducer.

### 2.4.5 OCR Combination

The combination of multiple recognition engines produces better results than single recognition systems. The early work of combining multiple outputs to correct OCR errors is described by Lopresti et al. [LZ97]. The method is called Consensus Sequence Heuristic Voting. In the method, the pages are scanned three times, and the OCR output is combined, then a voting procedure is applied. The recognition errors are produced by a single OCR package. The method is inspired from molecular biology. As the approach shows that dictionaries are able to insert characters, if none of the combined OCR systems produce them. Boschetti et al. [BRB+09] developed a workflow for digitizing Greek and Latin printed editions. It used Progressive Multiple Alignment method applied to different OCR outputs. The most similar pairs are aligned, necessary gaps are fixed to align the sequences and supplementary gaps are progressively added to the previous aligned sequences. The naive Bayes classifier is used to find the highest probability among multiple strings provided by the engines. Abdulkader et al. [AC09] reduced the error rate by using an error estimator neural network that learns to assess the error probability of every word from ground truth data.

The work of Lund et al. [LR11] proposed a decision list based on a combination of textual features of multiple OCR engines. The used features are based on voting values, word in dictionary, and other textual features where in-domain training data is available. Lund [LWR11] has also used progressive alignment to combine multiple OCR systems, then applied a supervised, discriminative machine learning method to select among multiple outputs. The Maximum entropy models are used and trained on synthetic data.

The Recognizer Output Voting Error Reduction (ROVER) [AAKK12] is a well known procedure for combining several speech recognition outputs. ROVER uses the frequency of occurrences and word level confidence values in the voting. However, those values cause poor voting outcomes due to the unreliability of the decoders confidence scores. Therefore, it is proposed that ROVER uses a pattern-matching-based voting scheme.

Wemhoener et al. [WYM13] proposed a method using Pairwise alignment and Multiple Sequence alignment to improve the OCR from multiple editions. The approach correct the OCR errors based on the majority choice.

The approaches can correct an error, if the correction occur in one of the OCR system. But they are unable to fix errors which have no correction in one of the OCR systems. A learning-based method is needed to vote for the best candidate corrections, when the multiple editions are unable to vote for the corrections, to resolve ambiguity in the choices, and when the correction is missing from the multiple editions so a prediction of unknown correction is required. Therefore, an LSTM corrections is designed in this thesis to solve all the above mentioned problems.

## 2.5 Toolkits and Open-Source Libraries for building Language Modeling

There are many tools which support the creation and evaluation of variety of LM's types based on n-gram statistics, as well as several related tasks, such as statistical tagging, manipulation of N-best lists, and word lattices.

Natural Language Toolkit (NLTK)[5] is a platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning. NLTK is available for Windows, Mac OS X, and Linux.

The MIT Language Modeling (MITLM) toolkit[6] is a set of tools designed for the efficient estimation of statistical n-gram LMs involving iterative parameter estimation. It achieves much of its efficiency through the use of a compact vector representation of n-grams. Details of the data structure and associated algorithms can be found in MITLM toolkit. It has been built and tested on 32-bit and 64-bit Intel CPUs running Debian Linux 4.0.

Speech Technology and Research Laboratory provided the SRILM[7] toolkit for building and applying statistical LMs, primarily for use in speech recognition, statistical tagging, segmentation, and machine translation. SRILM runs on UNIX and Windows platforms.

Carnegie Mellon Statistical Language Modeling (CMUSLM) toolkit[8] is meant for large amounts of training data. If you intend to train a LM from a few dozen or even hundred sentences. It is a set of Unix software tools designed to facilitate LM. It estimates word frequency lists, vocabularies, word bigram, trigram counts, various Backoff bigram, and trigram LMs.

In this section, we are also going to give an overview of several open-source toolkits which have been used for NLP application, and to represent the LM in Finite-State Transducers (FSTs). FSTs have key applications in speech recognition and synthesis, machine translation, OCR, pattern matching, string processing, machine learning, information extraction, and retrieval among others. Often a weighted transducer is used to represent a probabilistic model (e.g., an n-gram model, pronunciation model). FSTs can be optimized by determinization and minimization. The models can be applied to hypothesis sets and the best results can be selected by shortest-path algorithms.

OpenSFT [CAM07] is a library for constructing, combining, optimizing, and searching WFSTs. WFSTs are automata where each transition has an input label, an output label, and a weight. It provides the standard functionality which are required for WFSTs, such as compose,

[5] NLTK Toolkit: `http://nltk.org/`

[6] MITLM Toolkit: `https://code.google.com/p/mitlm/`

[7] SRILM Toolkit: `http://www.speech.sri.com/projects/srilm/`

[8] CMUSLM Toolkit: `http://www.speech.cs.cmu.edu/SLM_info.html`

closure, minimize, determinize, concat, prune, intersect, project, shortest path, remove epsilon, and union. It is powerful, widely used toolkits with optimized algorithms.

HFST[9] the Helsinki Finite-State Transducer software is intended for the implementation of morphological analysers and other tools which are based on weighted and unweighted FST technology. The feasibility of the HFST toolkit is demonstrated by (a full-fledged) open source implementation of a Finnish lexicon as well as a number of other languages of varying morphological complexity. OpenFST is used for providing the functionality for WFST technology.

XFST[10] Xerox finite-state tool is a general-purpose utility for computing with finite-state networks. It enables the user to create simple automata and transducers from text, binary files, regular expressions, and other networks by a variety of operations. The user can display, examine, and modify the structure and the content of the networks. The result can be saved as text or binary files. They can be combined by means of a variety of operations, such as union, and composition.

DFKI Finite-State Machine FSM toolkit [Pis02] is used for building, combining, and optimizing the Finite-State Machines (FSM). They provide standard operation for FSM with some other operations which include direct incremental construction of minimal deterministic acyclic FSM.

## 2.6 Discussion

In this Chapter, a summary of the state-of-the-art LMs in various application is provided. The standard LMs based on probabilistic n-gram models, with heuristic search to find the N-best path.

New LM techniques were invited using Neural Networks. The learning-based models showed good performance especially for the unseen tokens during the training.

This Thesis will show the improvement of the state-of the-art rule-based error model and also will show the outperforming LM using the learning-based model in various tasks.

Several open-source libraries are viewed to construct, evaluate the LM.

[9] HFST Library: `http://www.ling.helsinki.fi/kieliteknologia/tutkimus/hfst/`
[10] XFST Library: `http://www.cis.upenn.edu/~cis639/docs/xfst.html`

# Part I

# Language Modeling using Weighted Finite-State Transducers (WFSTs)

# Chapter 3

# Context-Dependent Confusion Rules for OCR Post-Processing

The correction of recognition outputs is an important first step in post-processing of recognition systems. The goal of improving recognition results is to convert a given document image into a machine readable format for books digitalization.

The designed error model transducers in this chapter are extracted from OCR confusions with context-dependent rules. The error model supports the Language Model (LM) to correct the words which are unknown in the dictionary and to reduce search time in dictionary. The error model and language model are represented in WFSTs.

The following contributions to the state-of-the-art in language modeling for recognition systems are presented in this chapter[1]:

1. Context-dependent confusion rules are used to build the error model.
2. They are based on the confusion matrix that the OCR produces and is dependent on the context of the strings (OCR results).
3. The context of the string is used to fit the confusion rule in the proper string where it belongs and brings the string to the corrections. It involves less rules in the composition which makes the search faster and accurate.
4. The size of the context rules is flexible.
5. The approach is language independent.
6. It is designed to deal with different number of errors.
7. It has no limited words size.

[1] This chapter is based on Al Azawi work in [AAB14a].

8. WFSTs are flexible, easy to adapt, and have fast computation.

9. WFSTs provide smoothed alignment between two different strings by allowing various types of transactions. Beside the input/output labels of the current string, the edit operations: insertion, deletion, and substitution can also be applied. Those operations are translated to context-dependent rules to help the transducers to compose the appropriate labels to choose the candidate outputs during the alignment. The weight supports to produce the best output from the candidate outputs.

10. Evaluation shows that the error rate of the implemented model on the UWIII testset is 0.68%, while the baseline is 1.14%, and the error rate of the existing state-of-the-art single character rules-based approach is 1.0%, as shown in Section 3.3.

11. The designed approach in the experiments requires 30 seconds, while the single-character error model takes 1 minute and 6 seconds.

It was proved by the research on language modeling for speech recognition [Moh03b] that the WFSTs are flexible, fast, and can be interpreted in different designs for lattices and string matching tasks. It is easy to align various WFSTs with the recognition input. The search is done by using optimized and high speed algorithms such as the OpenFST presented in [CAM07]. Therefore, the error model is designed using WFSTs, and not other structures.

The remainder of the chapter is structured as follows. In Section 3.1, the state-of-the-art single character error model is explained. Section 3.2 describes the designed method for building error models using context-dependent OCR confusion rules and WFSTs. The language model and the alignment are shown in Section 3.2.3. Section 3.3 presents the experimental results. The materials are explained in Section 3.3.1. Section 3.4 concludes the chapter.

## 3.1 Single Character Rules-Based Approach

The single character rules are extracted using Levenshtein edit distance algorithm [HNH08]. The rules represent the primitive operations: insertion, deletion, and substitution. These rules are used for constructing the transducers. Each transition in the transducer holds single rules, i.e., the substitution rule $c : a$, where $c$ is the wrong character in a given string, and $a$ is the correct character, as shown in Figure 3.1.
An example of insertion rule is $\varepsilon : v$ and deletion rule is $t : \varepsilon$. The transitions from the start state $S_{start}$ to the final state $S_{final}$ allow editing of the input strings according to the rules. The loop transitions on the start state $S_{start}$ and final state $S_{final}$ are called identity rules, and are used to pass the rest of the characters in the strings.

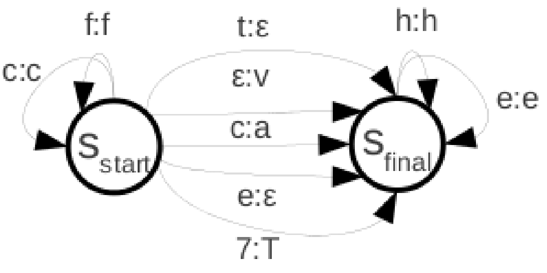


Figure 3.1: Sample of the Single Character Rules-based Transducer. The transitions from $S_{start}$ to $S_{final}$ are used to modify the strings such as: $t : \varepsilon$, $c : a$, and $7 : T$. The loop transitions hold the identity rules.

## 3.2 Designed Context-Dependent Error Model Method

In OCR systems the Language model consists of finite wordforms and their probabilities based on their occurrence in a given corpus, as shown in Section 3.3.2. The Language model can be constructed as deterministic FSTs which are fast and efficient to search in a set of words, and retrieve words. However, the search for candidate corrections for the misrecognized word in the language model is inefficient, and very slow for large-size dictionaries, because it requires a look-up over all the entries of the language model. Furthermore, the language model might fail to find the suitable candidate corrections for unknown wordforms. Therefore, a mechanism for generating correction suggestions for the erroneous wordforms is needed, which is called error model.

In this section, the extraction of the context-dependent rules, and construction of the error model transducer using those rules are described. The way that the language model is built, the alignment technique which is used, and the used materials are explained.

### 3.2.1 Context-Dependent Confusion Rules Extraction

The purpose of an error model is to act as a filter to revert the mistakes of the recognition outputs. The simplest and most traditional model for making such corrections is the Levenshtein-Damerau edit distance algorithm contributed initially by [Lev66b]. The correction of OCR errors using the confusion rules usually means generating a list of wordforms belonging to the given language.

The error model typically provides a small selection of the best matches for the language model to select from in relatively short time span, which means that when defining corrections, it is also necessary to specify their likelihood in order to rank the correction suggestions. The designed error model developes the Levenshtein transducer using the alignment of misrecognized words of the OCR output with their corresponding ground truth. By using the outputs of the alignment, the OCR confusions are extracted in a form of rules to be used in the error model with

Table 3.1: Context-dependent rules which are required to fix the misrecognized words and their correspondencing correct words.

| Context-Dependent Rules | Misrecognized Words | Correct Words |
|---|---|---|
| f$\varepsilon$n $\rightarrow$ fin | Defnition | Definition |
| f$\varepsilon$e$\rightarrow$ffe | efect | effect |
| m$\varepsilon$d$\rightarrow$mod | mdel | model |
| [l]$\rightarrow$[1] | [l] | [1] |
| Sqfecy$\rightarrow$Safety | Sqfecy | Safety |

respect to their context in both the misrecognized and ground truth wordforms. The extracted confusion rules represent the simple edit operations: insertions, deletions, and substitutions in a context-dependent form, as are shown in the examples in Table 3.1. The error model assigns cost for each rule. The cost is represented in weight $w$ to the confusions pair of the misrecognized word $r$ and its correct word $s$ corresponding to the probability of an OCR output word $r$ when intending to recognize the word $s$, and the context model assigns weight $w$.

The error model is built using edit distance algorithm; the misrecognition is assumed to be a number of operations applied to characters of a string: deletion, insertion, and substitution. With neighboring characters on leftmost and rightmost sides. Also the size of the context that are involved in the rule is controlled. To correct the first character, the context of the right side is taken. The context of the left side is taken to correct the last character. For example, the misrecognized word $Defnition$ needs the rules $f\varepsilon n \rightarrow fin$ to be fixed. The misrecognized word $efect$ requires the rule $f\varepsilon e \rightarrow ffe$, as are shown in Table 3.1.

### 3.2.2 Construct Error Model using Weighted Finite-State Transducers

In the next step, the error model transducer is constructed using the extracted context-dependent rules which are extracted in Section 3.2.1. The error model transducer is a weighted finite-state transducer that maps the misrecognized words to correct strings. Each of these context-dependent rules can receive a probability. The probability is derived from the confusion matrix of the OCR classifier. The context-dependent rules consist of two parts, the left part is the OCR confusions, and the right part is the corresponding ground truth. The rules are translated into a WFST, while the left part represents the input label of the transducer, and the right part of the rule represents the output label. The probabilities of these rules are taken as weights in the transducer.

Therefore, the error model transducer is able to map the OCR error by matching the output label of the OCR transducer with the input label of the error model. The output label of the error model is matched to the corresponding input label of the dictionary and maps the OCR error to its correspondence correction. For example, the rule $f\varepsilon c \rightarrow fic$ is highlighted in the

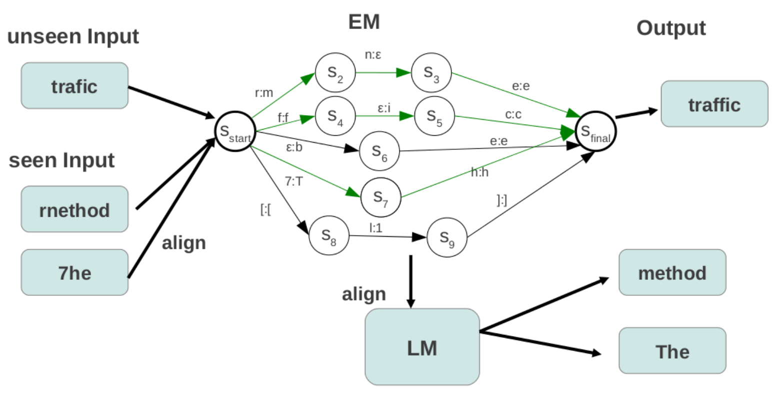


Figure 3.2: Sample of the context-dependent Confusion rules in Error Model (EM) transducer. The figure shows inputs/outputs examples for seen and unseen tokens in the EM and LM.

WFST as is shown in Figure 3.2.
The transducer starts from the start state $S_{start}$, the character $f$ represents an input and output label in the first transition to state $S_4$. The left part of the rule $\varepsilon$ is an input label in the second transition to state $S_5$, while $i$ represents the output label of this transition. Finally, from state $S_5$ to the final state $S_{final}$ the input, and output labels are $c$. The path from $S_{start}$ to $S_{final}$ is a successful path. The transition $\varepsilon : i$ means insert $i$ in the word which was missing from OCR outputs. Other identity rules, i.e., $a : a$, are also involved to pass the remaining characters of the string. Part of the constructed error model transducer is shown in Figure 3.2.

### 3.2.3 Language Model and Alignment Technique

The language model can be as simple as a list of finite words compiled into probabilistic WFST. The words and their frequencies were extracted from the text corpus from project Gutenberg[2]. The standard WFSTs framework is used to include probability estimates for constructing a unigram model.

In an n-gram model, the probability $P(w_1, ..., w_m)$ of observing the sentence $w_1, ..., w_m$ is approximated as:

$$P(w_1, ..., w_m) \approx \prod_{i=1}^{m} P(w_i \mid w_{i-(n-1)}, ..., w_{i-1}) \tag{3.1}$$

[2] Gutenberg Project: `www.gutenberg.org`

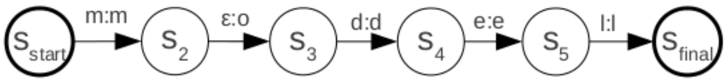


Figure 3.3: Output of the composition

The conditional probability is calculated from the n-gram frequency counts:

$$P(w_i \mid w_{i-(n-1)}, ..., w_{i-1}) = \frac{count(w_{i-(n-1)}, ..., w_{i-1}, w_i)}{count(w_{i-(n-1)}, ..., w_{i-1})} \tag{3.2}$$

It is assumed that the probability of observing the $i^{th}$ word $w_i$ in the context history of the preceding $i-1$ words can be approximated by the probability of observing it in the shortened context history of the preceding $n-1$ words.

In the next step, the three transducers are aligned:

$$OCR_{output} \circ EM \circ LM \tag{3.3}$$

Where $\circ$ represents the compose operation as described in Section A.1.1. First, the OCR outputs are aligned with the error model to generate composed Levenshtein transducer with OCR confusions of OCR output. The alignment technique is described in [AALB13]. It contains all the possible and appropriate candidate corrections of the misrecognized tokens, and provides suggestions to the language model to decide the best candidate correction which should be taken to correct the misrecognized tokens. Both the OCR output and the error model are represented in finite-state transducers.

After composing the input with the error model transducer, the resulting WFST is composed with the dictionary to filter out the words that do not exist in the language. Then, the best path with lowest cost is chosen from the second composition. A composed WFST will be generated with weights to be used in the ranking. The weights are based on the extraction which is described in the formula in Section 3.2.1. The composed WFST has many paths and depends on the compositions with the correspondence rules in the error model. The best path of composed WFST has the lowest cost and it is obtained by using a dictionary as a filter in the second composition. The best path is considered the output of the combination of the two compositions which include the composition of the OCR output, error model transducer, and language model transducer, as shown in Figure 3.3. The Figure shows the rule $m\ \varepsilon\ d \rightarrow m\ o\ d$ which is composed with the recognition output to correct the string *mdel* to *model*. In the second composition, the dictionary lookup will find the proper candidates corrections that are closed to the input string. As such an optimal and efficient searching in the dictionary is obtained with respect to time and space.

Table 3.2: Error rate of the designed Error Model compared to the original OCR recognition results and single character rule-based correction.

| Dataset | Baseline | Single Character Rules-based | Context-Dependent Confusion EM |
|---|---|---|---|
| UWIII-training | 1.28% | 0.79% | 0.10% |
| UWIII-testing | 1.14% | 1.0% | 0.68% |

## 3.3 Experiments

### 3.3.1 OCR and Materials

The OCRopus OCR system[3] is used to obtain the recognition of the University of Washington (UWIII) dataset [GHHP97] and separate the data into a training and a test set. The OCR outputs are extracted from lattices of character hypothesis. They are used in the testing and are transformed into finite-state transducers to be composed with the error model transducer. The implemented method does not impose any constraints on the bound $k$, where $k$ is the edit distance between the input word and the candidate corrections. The purpose of context-dependent error correction using OCR confusion is to rank a set of candidate corrections taking the misrecognized word context into account. In the implemented approach, there is no limitation or constraints on the length of the word or number of errors in the word.

The dataset is split into 6,000 training inputs and 3,917 testing inputs. The ground truth is used to evaluate the generated corrections wordforms. The dataset is generated from OCRed textlines of the UWIII datasets. The context-dependent confusion rules are extracted from OCRed pages of the UWIII dataset. The total number of the extracted set of rules in the error model are 162 rules. The number of entries in the language model is 144,240 and generated from project Gutenberg[4]. The number of rules in the single-character model is 208. The number of identity rules is 78, for example, $e \rightarrow e$.

### 3.3.2 Experimental Results

In the experiments, OCR output and the corresponding ground truth is used to build the error model. Table 3.2 shows the evaluation of the implemented error model approach compare to the original OCR recognition results and the state-of-the-art approach single character rule-based [HNH08].

The Character Error Rate (CER) is measured using the edit distance to find the number of

[3] OCRopus Project: `https://github.com/tmbdev/ocropy`
[4] Gutenberg Project: `www.gutenberg.org`

Table 3.3: Output examples of the designed error model compared to the original OCR recognition results.

| Misrecognized Words | Correct Words | Applied Context-Dependent Rules | Edit Operations |
|---|---|---|---|
| $nefer$ | $refer$ | $ne \rightarrow re$ | substitution |
| $peEformed$ | $performed$ | $eEf \rightarrow erf$ | substitution |
| $AUsTIN,$ | $AUSTIN,$ | $UsT \rightarrow UST$ | substitution |
| $7he$ | $The$ | $7h \rightarrow Th$ | substitution |
| $rnethod$ | $method$ | $rne \rightarrow m\varepsilon e$ | deletion and substitution |
| $artifcial$ | $artificial$ | $f\varepsilon c \rightarrow fic$ | insertion |
| $ecause$ | $because$ | $\varepsilon e \rightarrow be$ | insertion |
| $diHerent$ | $different$ | $iH\varepsilon e \rightarrow iffe$ | insertion |

edit operations on character level.

$$CER = \frac{I + D + S}{N} \times 100 \tag{3.4}$$

where N is the total number of characters. I is the number of insertions, D is the number of deletions, and S is the number of substitutions. The experimental runtime for the implemented context-dependent error model was 30 seconds, while the single-character error model took 1 minute and 6 seconds for the testing set.

An example, the confusion rule $p\varepsilon r \rightarrow per$ is used to fix the misrecognized word *papr* to *paper* by insertion of the character $e$. Another example is transforming the misrecognized word $lt$ to the correct word $It$ using the rule $lt \rightarrow It$ by substituting the character $l$ with $I$. More examples are shown in Table 3.3. The misrecognized word *ecause* is corrected by using the context-dependent rule $\varepsilon e \rightarrow be$ to *because*, while the result using single character rules is *cause*. The unknown misrecognized word $trafic$ overlaps with the rule $f\varepsilon i \rightarrow ffi$ and the alignment produces the correct word $traffic$. While the misrecognized word $Transportation$ is not corrected $Transrtation$.

An example of a confused case, i.e., a misrecognized word $a$ is considered as a dictionary word but in the original context is different word *and*, which it misrecognized by the OCR as $a$.

The error model is able to find the correspondence confusion rule while the language model considers it as a correct word. Finally, the best path with the lowest cost is chosen from n-best paths. The implemented error model is capable of correcting any type of errors such as transform an incorrect word to a dictionary word or transform the wrong symbols to its correspondencing digits or punctuations. The implemented model learns the OCR confusions and converts them to rules to edit, and correct the strings. The size of the context that is used in the rules is flexible. If the word is not in the dictionary, then only the error model is used. The error model can fix any number of errors per word. The length of the words is not fixed. Words with different length are applied.

## 3.4 Discussion

In this chapter a WFST-based method is used to build an error model to correct OCR errors. The error model is established from the context-dependent confusions of the OCR errors and is constructed using WFSTs. The context-dependent confusions are extracted using the Levenshtein edit distance alignment. The Levenshtein edit distance algorithm is used to construct the error model. The primitive operations, such as deletion, insertion, and substitution are used to fix the misrecognized tokens. A number of edit operations are applied on the character in the string with respect to the leftmost side or rightmost side of the sting. The rules cover the OCR confusions.

Using the technique from the Finite-State theory and avoiding calculating the edit distance makes the approach fast and efficient. The new implemented method avoids calculating the edit distance at all steps of the correction process. It helps the language model to find the candidate corrections efficiently and with a controlled search. The designed method uses the context of the string to extract the rules and later the rules will be more efficient and convenient to fit in, and correct the wrong string. The implemented model is capable of correcting any type of errors such as transforming an incorrect word to a dictionary word, or transforming the wrong symbols to digits, and any other errors.

The implemented error model is fast, efficient, and performs better than the state-of-the-art approaches [HNH08, LNCPCA10]. The single character rule-based method required a lot of effort to obtain an optimized model and accurate results. Tuning and pruning with different parameters are needed to solve the delay in composition, but might not lead to accurate results. The single rule-based approach is slow because it involves many irrelevant edit operations in the composition between the input and the error model. The state-of-the-art in spell-checkers showed that the rules-based approach is better to aid the language model, while alone language model is not capable of doing the corrections, moreover language model are often slow. However, the single character rules-based requires to be optimized [HNH08]. The parameter for smoothing the weights needs to be tuned to have optimized costs. The composition with such a model needs to use more operations such as pruning [LNCPCA10] and remove epsilon. The single character model needs to reduce the usage of epsilons and identity transitions to avoid the delay in the composition. Their search is not restricted and involves in irrelevant and unsuited candidate correction or transitions which makes the composition slow and the WFST huge. However, such transitions with epsilons, and identity rules are necessary in the composition. Using an inappropriate threshold during the pruning process on the transducers leads to a loss of the best path with low cost. Such problems were not faced in the designed context-dependent scenario. Therefore, the designed context-dependent error model is suggested which shows the improvement on the OCR results compared to the single character rules.

The implemented WFSTs approach has no limitation on the words length and the number of the errors that the method should correct. The experimental results show that the implemented method achieves good performance for both time and correction accuracy. The approach is completely language independent and can be used with any language that has a dictionary and text data to build a language model.

Chapter 4

# Transcription Alignment

The motivation of this chapter is two-fold. First, available manual transcriptions in standard edition lack the possibility of linking from the transcription to the corresponding positions in the original document, and second, a huge amount of training data with aligned transcriptions would be necessary in order to create good OCR systems for historical documents. The typical manual transcribing process usually provides different editions of transcriptions which are nowadays available for some manuscripts. Originally, transcriptions were created for just assessing the contents of the text rather than for providing an alignment between the transcribed edition, and the original image. Hence correspondence at the word, line, or sentence level may not be available. This is especially true for historical documents[1]. Therefore training and evaluation of techniques for recognition and retrieval of historical documents with different levels of text modification is still a big research challenge.

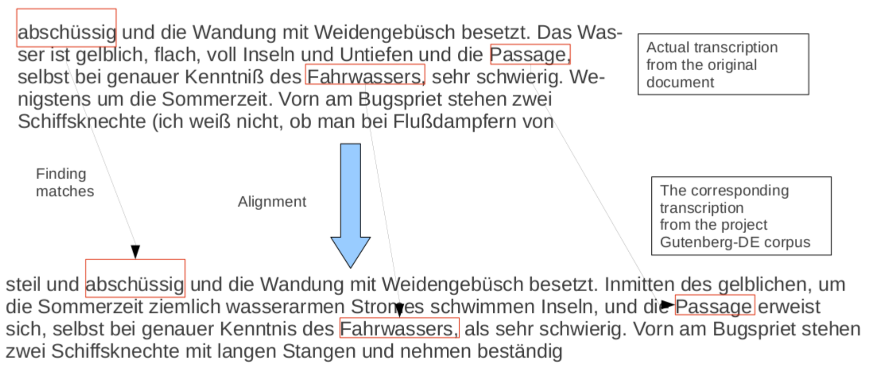
The first part describes the implemented approaches for transcription alignment with the hyphenation model using WFSTs. The second part presents the dehyphenation for noisy transcription is by classification.

C1: The first part of the chapter discusses WFST-based alignment approaches which are scalable and are well suited for historical documents, as shown in Section 4.2. The purpose of Section 4.2 is to provide an easy-to-use implementation with no parameter tuning but still yielding lower error rates. In particular, the contributions are divided into two scenarios in Section 4.2.

(a) Alignment approaches are designed under OCR layout and recognition errors with desired transcription.

(b) Aligning the OCR lines with imperfect transcriptions which have different text or missing contents, words with different upper or lower case letters, and different punctuation.

[1] This chapter is based on Al Azawi works in [AALB13, AAB14b].


abschüssig und die Wandung mit Weidengebüsch besetzt. Das Was-
ser ist gelblich, flach, voll Inseln und Untiefen und die Passage,
selbst bei genauer Kenntniß des Fahrwassers, sehr schwierig. We-
nigstens um die Sommerzeit. Vorn am Bugspriet stehen zwei
Schiffsknechte (ich weiß nicht, ob man bei Flußdampfern von

steil und abschüssig und die Wandung mit Weidengebüsch besetzt. Inmitten des gelblichen, um
die Sommerzeit ziemlich wasserarmen Stromes schwimmen Inseln, und die Passage erweist
sich, selbst bei genauer Kenntnis des Fahrwassers, als sehr schwierig. Vorn am Bugspriet stehen
zwei Schiffsknechte mit langen Stangen und nehmen beständig

Actual transcription from the original document
Finding matches
Alignment
The corresponding transcription from the project Gutenberg-DE corpus

Figure 4.1: Text lines from the actual OCR transcription from the original document and the corresponding transcription from the project Gutenberg-DE corpus. Each word in the actual transcription must be aligned with the corresponding word in the other text. The transcription from Gutenberg-DE corpus has different contents, line breaks, and no hyphenation

(c) Typically, the transcription has different line or page breaks which means having no information about the correspondence at the word, line, sentence level or page level. The implemented approaches are robust and able to solve the correspondences problem.

(d) A novel hyphenation model is designed using WFST to adapt any unhyphenated transitions or transitions with different hyphenation differ from the hyphens in the OCR's lines.

C2: The second part of the chapter is the dehyphenation for noisy transcriptions and is described in Section 4.3.

(a) The purpose of dehyphenation is to provide the output of the text recognition systems in a clean format by removing hyphens which are used as line breaks.

(b) Furthermore, it removes all the misrecognized hyphens,in order to deliver a clean, and easily adaptable text format of the recognition results to be used for different purposes.

The remainder of the chapter is outlined as follows. Section 4.1 describes the alignment problem. Section 4.2 describes several WFST-based alignment approaches for OCR errors and layout variations for historical language. The designed approaches page-wise alignment in Section 4.2.1 and small text segment alignment in Section 4.2.2 are compared with the state-of-the-art book-wise approach [YM11] in Section 4.2.3. Section 4.3 describes the dehyphenation of noisy transcriptions. Section 4.3.1 shows the dehyphenation problems in the noisy transcriptions and Section 4.3.2 describes the challenges in English and German.

The implementation of the designed methods for de-hyphenation is described in detail in Section 4.3.3. Section 4.4 presents the experimental results. Section 4.4.1 presents the transcription alignment approaches and Section 4.4.2 shows the results of dehyphenation approaches. Finally, Section 4.5 concludes the chapter with a discussion.

## 4.1 The Alignment Problem

Historical books may contain different fonts, touching, broken characters, and warped characters at the edges. Furthermore, the scanning process often introduces blur and curved lines, and there are numerous other possible sources of noise like bleed-through, degraded character, and annotations [BC11, NSD+11, LdFPeSdAF11, MEAMA11, FV11]. An example font is Fraktur, which is a form of black letter or Gothic script, and includes the elongated s, and ligatures, or "joined" letters for certain letter combinations. Fraktur documents have multi-column layout with narrow line spacing which leads to touching lines. Bad scanning resolution or degraded documents lead to many segmentation and recognition errors. Applying OCR on those images might produce many words with at least one or more wrong characters in the word, which might disturb the idea of finding unique words, and producing a long segment for the alignment as was shown in Manmatha et al. [YM11]. An important task is creating a correspondence between the OCR results and a differ ASCII text of the document. To be able to align the erroneous OCR lattices with a layout variable ground truth, an adapted transcription model is designed using WFST with edit operations such as addition, deletion, and substitution. With this approach there is no need to revise the numerous wrong characters or words in the the OCR transcription before the alignment.

The transcriptions were taken from project Gutenberg-DE[2] which has different line breaks, page breaks, hyphenenation, and punctuations. Furthermore, the capitalization of the words in the text edition do not correspond to the images of these words. The transcription includes missing text lines and also has different text content from the image, as shown in Figure 4.1.

## 4.2 WFST-Based Transcription Alignment Methods

Aligning the OCRed text line lattices directly to transcriptions might not align the pairs correctly. An important reason for that is that the sequence of the OCR lines is not the same as in the original document or the transcription. Other factors which can affect the alignment are insufficient information at word or line level of the transcription. Therefore, different alignment approaches are implemented using WFST to avoid mismatched parts between the OCR lattices and the transcriptions, and guarantee the OCR lattices will be aligned to the correct text contents in the transcriptions to get as much as possible of those alignment pairs. In the following

[2] Gutenberg Project: `www.gutenberg.org`

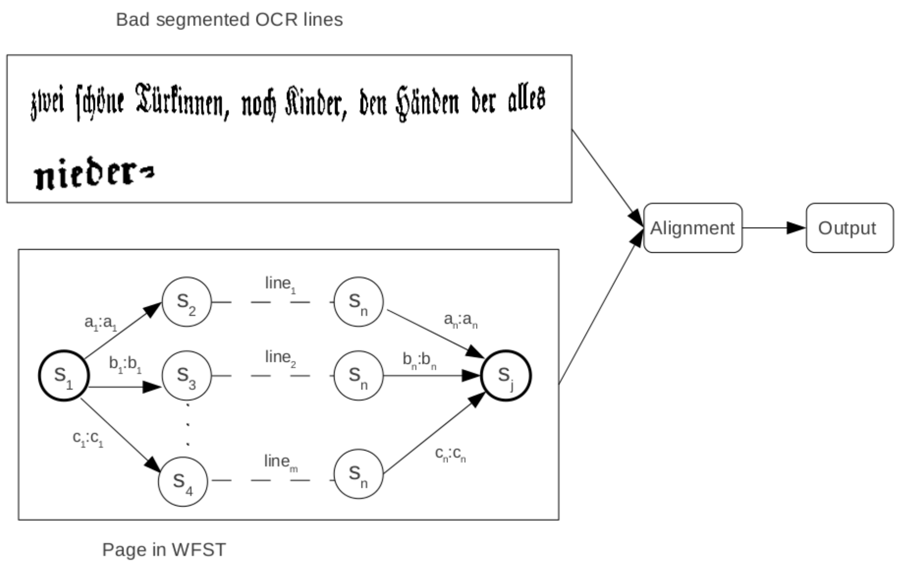


Figure 4.2: Page-Wise WFST-Based Alignment Approach. The Page-Wise alignment approach aligns each OCR lines with the page transcription (on the right side in WFST) separately, while the text line has a segmentation problem.

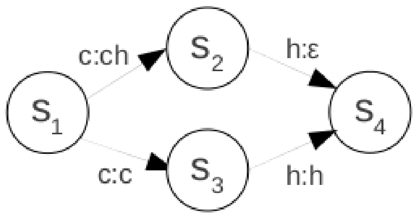
sections the designed alignment approaches are presented.

### 4.2.1 Page-Wise Approach

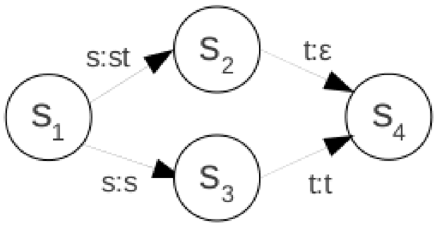
To avoid the problem of mismatched sequences, a page-wise approach is designed as parallelized text lines in an FST. The approach is used in aligning an OCRed text line with a line from the transcription, while it is unknown, if the sequence (order) of the OCRed text lines are the same as in the transcription. The difference in the line order comes from segmentation errors or transcriptions having different line order or line breaks. Figure 4.2 shows the idea of the page-wise alignment approach which aligns separately each OCR line lattices with the whole page transcription.

A page-wise approach is designed to avoid the differences in the lines' order in the OCR from the transcription. Each line in the transcribed page is represented as a line in the FST, altogether they are represented as parallel lines for the page FST. This approach aligns the line lattice of the OCR with each line in the page FST, then the best valid path is chosen from the composed graph which represents the best match between the OCR and the transcription. Figure 4.2 shows how the lines in a page as FST are represented. The page FST consists of "$j$" states and "$m$" lines.

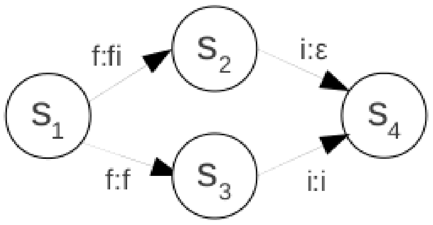
Each line is represented as a string and each character is considered as an input and output label for the transition between every two states. Considering "$a_1$" as the input and output



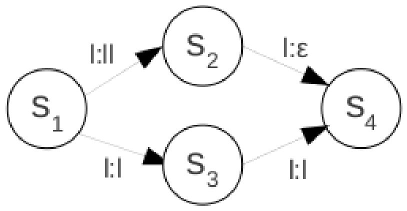
(a) Fraktur ligatures *ch*



(b) Fraktur ligatures *st*



(c) Fraktur ligatures *fi*



(d) Fraktur ligatures *ll*

Figure 4.3: Presentation of the Fraktur ligatures in FST: "*ch*", "*st*", "*ll*" , and "*fi*".

label for the first character in the line, this representation is followed until the last character in the line which is "$a_n$" is reached. Each line in the transcription is presented as WFST line and starts from the start state and ends in the final state. The same representation is followed for the next lines until the end of the last line. The combined WFST lines generate the WFST page of the transcription. All the WFST lines on one page, start in the same start state, and end in the same final state.

Segmentation of Fraktur characters is a challenge because the Fraktur documents include many touching characters which are represented in the clustering and recognition as combined characters or so-called ligatures. The OCR lattices include many combined characters as one segmentation. Therefore, those characters are also represented as single characters, and/or as combined characters if they are considered as Fraktur ligatures to find the suitable match pairs in the alignments. In Figure 4.11(a), the representation of the ligature "*ch*" is shown, which is for part of the the word, for example, "*machen*". The transition "$c : ch$" has "$c$" as an input label and "*ch*" as an output label, the second transition "$h : \varepsilon$" has "$h$" as an input label, and "$\varepsilon$" as an output label to produce the "*ch*" as ligatures, which is a typical Fraktur ligature and appears often. The parallel transition is present in the FST to produce the "$c : c$" and "$h : h$" respectively, if the ligature does not occur in the OCR lattices. This design is followed for most frequent Fraktur ligatures.

The edit operations substitution, insertion, and deletion are used in the approach to smooth the alignment between OCR and the transcription. They are translated as transitions in the FST and used as rules to substitute, insert, and delete characters from the OCR. Those

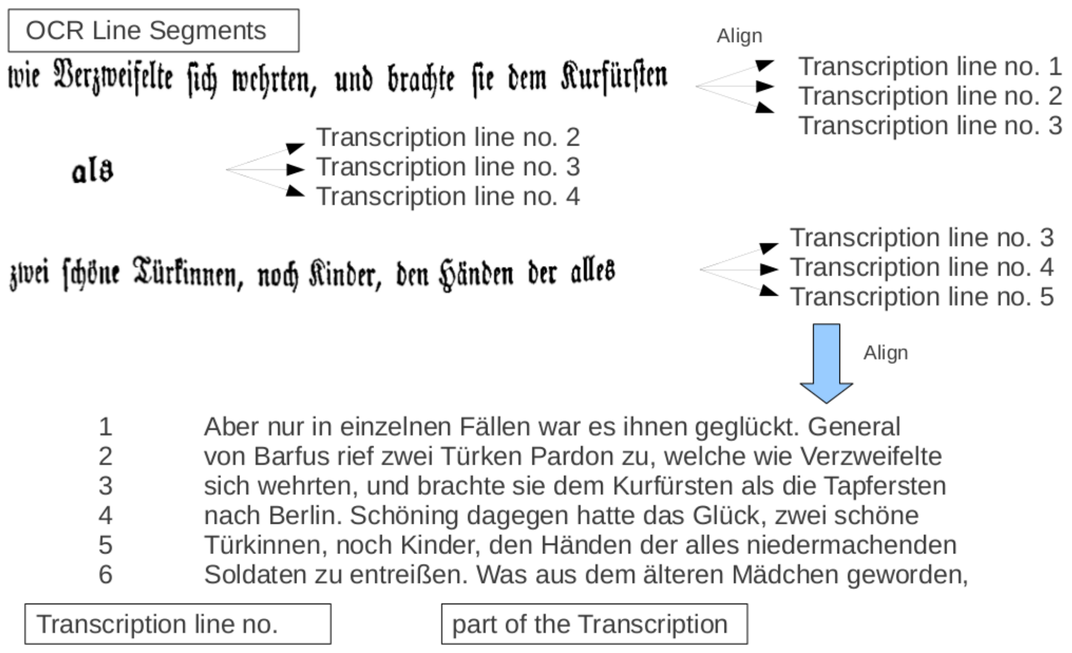

OCR Line Segments
wie Verzweifelte ſich wehrten, und brachte ſie dem Kurfürſten
als
zwei ſchöne Türkinnen, noch Kinder, den Händen der alles
Align
Transcription line no. 1
Transcription line no. 2
Transcription line no. 3
Transcription line no. 4
Transcription line no. 5
1 Aber nur in einzelnen Fällen war es ihnen geglückt. General
2 von Barfus rief zwei Türken Pardon zu, welche wie Verzweifelte
3 sich wehrten, und brachte sie dem Kurfürsten als die Tapfersten
4 nach Berlin. Schöning dagegen hatte das Glück, zwei schöne
5 Türkinnen, noch Kinder, den Händen der alles niedermachenden
6 Soldaten zu entreißen. Was aus dem älteren Mädchen geworden,
Transcription line no.
part of the Transcription

Figure 4.4: Alignment of the OCR line against the Transcription lines.

editing rules are: Substitution: "$\rho$" $\rightarrow$ "$c$", Deletion: "$\rho$"$\rightarrow$ "$\varepsilon$", and Insertion: "$\varepsilon$" $\rightarrow$ "$c$", where "$c$" is a character, "$\rho$" is Rho and "$\varepsilon$" is epsilon. The Rho matcher is used if there is no symbol in the OCR Lattices that match the transcription characters. More generally, they may implement matching special symbols that represent sets of labels such as "$\rho$" (rest), which can be used for more compact automata representations, and faster matching. For each character in the line, the designed edit rules are applied. The cost of those edit operations is empirically estimated. Matchers can find and iterate through requested labels at WFST states; their principal use is in composition matching. In Figure 4.3 ligature representations with the regular representation of the character itself are shown. They are then combined in one WFST as shown previously in Figure 4.2. The punctuation and special symbols are included as it is written in the transcription.

### 4.2.2 Small Text Segment Approach

To avoid the problem of different line breaks in the transcription and the over-segmentation problem in the OCR, an alignment approach is designed to find matches of the OCR lines (broken or correct line) in small parts of the transcription. A segment of text lines is extracted from the transcription and is defined as being less than a page and containing a text line from the transcription, and its surrounding lines, i.e., $n$ previous and $m$ next lines, where $n$ and $m$ typically range from 2 to 5. Figure 4.4 shows OCR lines having an over-segmentation problem

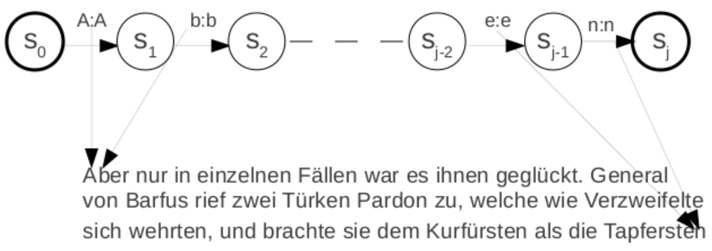

A:A
b:b
e:e
n:n
S0
S1
S2
Sj-2
Sj-1
Sj
Aber nur in einzelnen Fällen war es ihnen geglückt. General
von Barfus rief zwei Türken Pardon zu, welche wie Verzweifelte
sich wehrten, und brachte sie dem Kurfürsten als die Tapfersten

Figure 4.5: Extracted text lines segment of length "$j$".

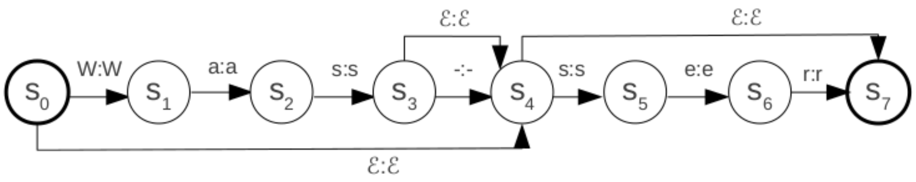


Figure 4.6: The word "Wasser" represented in FST with hyphenation.

and the transcription having different line breaks.

The desired line is extracted with the surrounding lines, for example, here a text segment with three lines is extracted. The extraction of the surrounding text line for the segment can be increased according to differences in the lines order and breaks between the OCR, the transcription, and the number of over-segmentations in the OCR. This approach can align text lines and solves the problem of mismatches as in the page-wise alignment. The text lines segments are represented as line WFST as shown in Figure 4.5. The first transition has the first character in the segment and the edit rules are used for each character, as shown in Section 4.2.1. The last transition represented is the last character in the segment. Each OCR line lattice is aligned with the text lines segment. The ligatures are also considered in the FST representation as in Figure 4.3.

Each combination of characters in the transcription is represented with the help of a big list of Fraktur ligatures. If it matches a ligature in the list. Skipping rule is added by using the epsilon transition to skip those symbols in the transcription which do not match any symbols in the lattices.

Noteworthy, this model is applied with and without hyphenation. The purpose of the hyphenation model is to hyphenate the words in the transcription which have no information about line breaks and the hyphenated words. Therefore, each word in the transcription is hyphenated. Then, the words are presented within the text lines segment FST. Figure 4.6 shows the designed hyphenation model which can produce the word water without hyphen

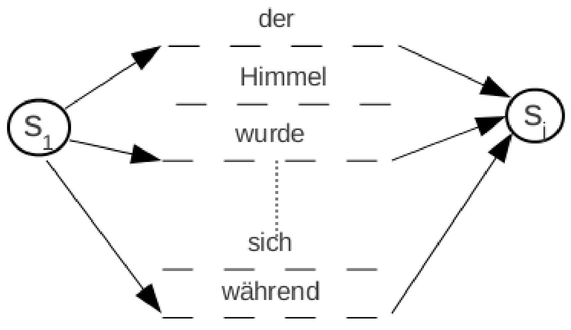
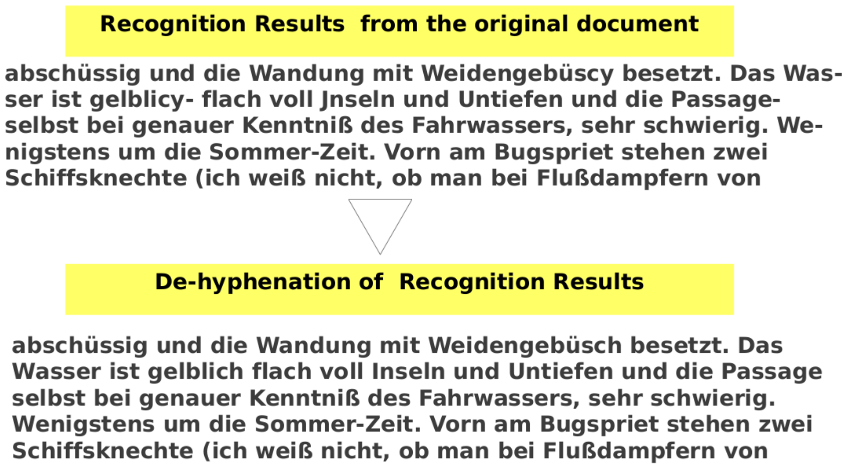


Figure 4.7: Book-Wise N-gram WFST-Based alignment approach.

"*Wasser*" or the hyphenated "*Was–*" and "*ser*" with certain cost. This model can produce the required word or part according to the OCR token.

### 4.2.3 Book-Wise N-gram Approach

The idea of this approach is aligning the OCR line lattices with the whole transcription content which is a book. The work of Manmatha et al. [YM11] aligns the whole OCR text output with the whole transcription and finding the unique words which help to clarify the correspondences between words in the OCR and the transcription. This idea is applied for evaluating the OCR results. The purpose is to generate the aligned pairs from the character segments of the manuscript and transcription for training purposes. Instead of getting the best path from the OCR lattices and matching it against the transcribed book. The corresponding text segment is then aligned with the OCR lattices. A simple approach is implemented to align the OCR lattices against the whole transcription directly using Weighted Finite-State Transducers. The WFST can find the correspondences on character level and is easier and simpler to use to generate the aligned pairs. This idea is supported when there is fuzzy information about the page breaks and big differences in the pages content between the OCR, and the transcription. Especially when the OCR produces a lot of errors due to the different fonts, touched, and broken characters and bad scanned historical documents. Figure 4.7 shows the book-wise based WFST. The WFST consists of parallel paths each path represents a word of the transcription. For each word, the normal and ligature characters representation are used, as shown in Section 4.2.2. Again, the hyphenation model described in Section 4.2.2 is integrated into this model. In the experiments, this approach is applied with and without hyphenation.


**Recognition Results from the original document**

abschüssig und die Wandung mit Weidengebüscy besetzt. Das Was-
ser ist gelblicy- flach voll Jnseln und Untiefen und die Passage-
selbst bei genauer Kenntniß des Fahrwassers, sehr schwierig. We-
nigstens um die Sommer-Zeit. Vorn am Bugspriet stehen zwei
Schiffsknechte (ich weiß nicht, ob man bei Flußdampfern von

**De-hyphenation of Recognition Results**

abschüssig und die Wandung mit Weidengebüsch besetzt. Das
Wasser ist gelblich flach voll Inseln und Untiefen und die Passage
selbst bei genauer Kenntniß des Fahrwassers, sehr schwierig.
Wenigstens um die Sommer-Zeit. Vorn am Bugspriet stehen zwei
Schiffsknechte (ich weiß nicht, ob man bei Flußdampfern von

Figure 4.8: Goal of the dehyphenation approach: the recognition result is the input and the recognition without the line breaks and noisy hyphens.

## 4.3 Dehyphenation of Historical Noisy Transcriptions by classification

The main aim of dehyphenation is to remove the hyphen and convert the result of OCR into a readable and reflowable format, so that they can be used for different purposes such as post-processing, and digitalization of the transcriptions with the corresponding historical books. The documents emphasize that hyphens are not only used as line breaks. For languages, such as German, hyphens are frequently used to combine two nouns to produce a word.

The remainder of this section describes the problems in Section 4.3.1, the challenges in Section 4.3.2. The implementation of the implemented methods for dehyphenation are described in details in Section 4.3.3.

### 4.3.1 Dehyphenation Problems in Noisy Text

This section describes the dehyphenation problems in noisy transcriptions of German and English script, as shown in Figure 4.8. The transcription is called noisy because it is obtained either from recognition systems with errors or when the users are typing text with various spelling and typing errors. To dehyphenate the noisy transcriptions, a robust method against spelling and recognition errors is required. Misrecognition example are substitution of symbols or dashes with hyphens.

A requirement is the ability to distinguish hyphens at the end of the line from those of compound words. The purpose is to classify the hyphen as a hyphenation of line break or hyphenation of combined words and terms. Furthermore, the transcription has a lot combined digits and letters, not only words, with a hyphen such as: “NS-1039-11”, and “l073-l078”.

Table 4.1 shows the recognition errors occur with hyphens. Compound words appear in three

Table 4.1: Misrecognitions with hyphens

| Error Types | Tokens |
|---|---|
| Recognition Error *w* | wHOLE-BODY |
| Insertion Error of Hyphens and Deletions | high-r-lution |
| Recognition Error $\sim$ | Go$\sim$back-N |
| Insertion Error of Hyphens | Schlachtfeld-,Während |
| Recognition Error | Witioenschteiev-1o) |
| Insertion Error of $-$ and , | Untergenerm-,, |
| disambiguities | I-\n je |

tokens “Electro-Magnetic-Acoustic”, line breaks hyphenated words such as “distri-\n bution”, and words which are hyphenated at the line break “plystyrene-poly-\n methylmethacrylate”. The recognition errors appear also in the compound words “Spät-NachmittaM-Beleuchtung;”, “Werbelliner-Forstj”, “Friedland-1788–180;j”, “16-Jahrhuntertg”, and “literarisch-aes)hetischer”. In the Fraktur documents, few compound words could appear with no capitalization e.g., “wiffenschaftlich-praktisch”. In the scanned books, the hyphen could appear at the line break of the page’s last line. In the next page, the page number is given, and then the rest of the hyphenated word.

### 4.3.2 Language-Specific Challenges

This section describes the specifics of English and German hyphenation which appear in clean and noisy texts.

In English, hyphens are used to link words and parts of words. They are not as common today as they used to be, but there are three main cases where they should be used: in compound words, to join prefixes to other words, and to show word breaks.

1. Hyphens are used in many compound words to show that the component words have a combined meaning (e.g., “a pick-me-up”, “mother-in-law”, “good-hearted”) or that there is a relationship between the words that make up the compound: for example, “rock-forming minerals are minerals that form rocks”. Compound adjectives are made up of a noun and an adjective: “sugar-free”, a noun and a participle: “computer-aided”, or an adjective, and a participle: “quick-thinking”.

2. Hyphens can be used to join a prefix to another word, especially if the prefix ends in a vowel, and the other word also begins with one (e.g., “pre-eminent” or “co-own”).

3. Hyphens showing word breaks. They show where a word is to be divided at the end of a line of writing, for example, “hel-met” not “he-lmet”.

The hyphen (der Bindestrich) in German is used very much like in English. Some of its lesser used functions are:

1. To indicate a link between syllables when lack of space at the end of a handwritten/printed lines forces the writer to separate a word.
2. To indicate a common suffix to words: “saft- und geschmacklos” (not juicy and without taste).

In German specifically, hyphens can be used to combine (nouns,nouns), (adjectives,adjectives), (suffixes,nouns), and (suffixes,adjectives), plus much more. Here are the rules for hyphenation in German as it currently stands:

1. The users have the choice to use the hyphen in: any long compound word where a hyphen would improve its readability and clarity “Kundenauftragsabwicklungsprozess” to “Kunden-Auftragsabwicklungsprozess”. With foreign words, namely the many English words that have infiltrated German business and media lingo “Job-Share” or “Jobshare”. However, foreign words will be written together when the first word cannot stand alone as an actual word, such as “der Afrolook”, “der Neofaschismus”. Compound words that contain three of the same letters one after another, such as “der Kaffeeersatz” to “der Kaffee-Ersatz” (coffee substitute).
2. Hyphens must be used when combining: numbers and words other than“-fach” and “Jahr”, for example “16jährig” to “16-jährig”(16 year-old), “5mal” to “5-mal” (5 times). Abbreviations and nouns: “Der Lkw-Fahrer”(truck driver), “die UV-Bestrahlung” (UV radiation). Single letters and words: “Das T-Shirt”, “die U-Bahn”. Words can combined with or without hyphens, e.g., “Ticket- und Tarif-Infos” (Ticketing and rate information) and “Bus- und Bahnfahrpläne” (bus and train timetables).

### 4.3.3 Designed Classification-based Methods for Dehyphenation

In the designed approach, the dehyphenation is treated as a classification problem. Hyphens are spaced apart enough so that each hyphen/no-hyphen decision is treated as a separate problem.

#### Feature Extraction

It is important to select appropriate features that can be learned for building a dehyphenation model based on the data in training sets. How those features should be encoded, which features are most informative, and how those features relate to one another. The appropriate features which are extracted from Fraktur are word length and the length of both tokens joined by a hyphen because German words are longer than a hyphenated part. Capitalization is one of the most appropriate features because the second part of the compound word could start with capital which makes it distinguishable. German nouns start with capital letters, it is more frequent that a word appears after a hyphen as a second part of the compound word than a second part of a hyphenated word at the next line [GT94].

Other features are the length of the two parts of either a hyphenated word or a compound word. It is necessary to find some other features to be used with the mentioned features for the English input. An example, part-of-speech tagging is used to distinguish between words, and hyphenated token because only words can have tags. Most compound words in English start with lowercase and some compound words have a short length. Therefore, the part-of-speech tagging is used. The two tokens which represent a compound word can have a POS-tagging, since it could be a combination of noun-noun, adjectives-adjective, noun-participle, and adjective-participle. However, the two tokens which represent a hyphenated word, they are not dictionary words, and cannot have POS-tagging. The hyphenated words in English are divided into syllables, which have no POS-tagging. The NLTK available tagger is used which is based on the Penn Treebank tagset [LB02]. For noisy transcriptions, a tagger supposes to work under OCR's errors to find the right tagger for the each hyphenated tokens. The tagger determines the tagging by checking the end of the word, but the tagger should avoid tagging a token such as "istry" in "chem-istry" by using a regular expression tagger.

The hyphen divides the words between syllables, e.g., "bas-ket","pic-ture". It can be avoided to carry over two-letter syllables to the next line, e.g., "fully", not "ful-ly". It can not divide a word of one syllable or divide any word that will result in a single-letter syllable, e.g., "again", not "a-gain". The hyphen also divides between double consonants, e.g., "equip-ping", not "equipp-ing". Therefore, the feature vowel is used to learn the characteristic of the syllables.

The frequency of the input which occurs either as hyphenated word or compound word is also used with the other features. In English, there are some cases where there is no distinguishable difference between the length of the compound words, and normal words. Furthermore, the compound word does not start with capitalization. The features set is fed into the model, which generates predicted labels. The input features are:

- The word length: the length of the two tokens which connect with the hyphen.
- The length of the first part of the word: the length of the first part of the hyphenated or command word.
- The length of the second part of the word: the length of the second part of the hyphenated or command word.
- The word has a Tag: binary ("yes" or "no") and provided by checking the the tag of the two tokens. The value is "yes" if first and second token have POS-tagging.
- Check Vowel: binary ("yes" or "no") and provided by checking the last syllable of the first token.

Table 4.2: Performance evaluation results of the alignment approaches applied on original transcription.

| Alignment Approaches | Total Number of obtained Characters | Error Rate |
|---|---|---|
| Small Text Segments | 137,408 | 0.6% |
| Page-Wise | 111,572 | 3.2% |

- Check Vowel: binary ("yes" or "no") and provided by checking the first syllable of the second token.
- Has a capital: binary ("yes" or "no"). The value is "yes" if the second token starts with upper case letter.
- Frequency: an integer value. It refers to the frequency of occurrence of the tokens within some given text corpus.

For classification, Decision Trees, Naive Bayes, and Maximum Entropy [LB02] are used.

## 4.4 Experiments

### 4.4.1 Experimental Results of WFST-based Transcription Alignment

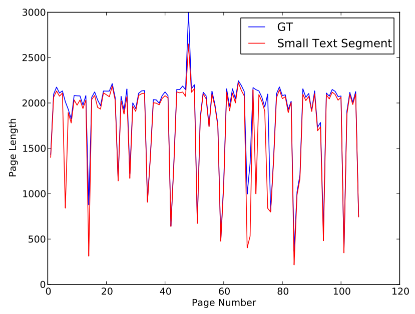
The alignment approaches are evaluated on Fraktur documents from real scanned books. The volumes of "Wanderung durch die Mark Brandenburg" (1862-1889) of Heinrich Theodor Fontane are used which have been provided from the German Text Archive (DTA). The recognition lattices are obtained by using open sources OCR system OCRopus [Bre08]. In the first part of the experiments, the approaches are applied on the original real transcriptions (ground truth) to assess their performance under OCR segmentation and recognition errors. The methods are applied on 106 Fraktur documents which have 3353 text lines and the results are shown in Table 4.2.

The first approach of small text segments returns 137,408 single and combined (ligatures) characters. The number of the returned ligatures is 6,941 which might be a combination of two, three or four characters. Those characters are stored in a database as tuples representing the correspondence between the segmented character(s) (of the manuscript) and the transcription character(s) which it has been aligned with. The error rate was 0.6%. The page-wise approach returns 111,572 characters with error rate 3.2% and 587 ligatures.

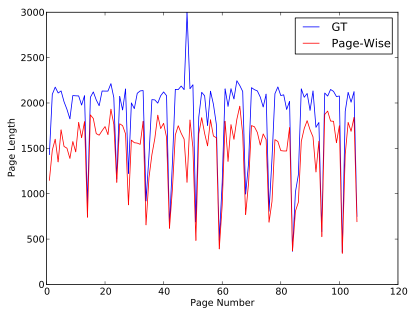
The error rate is measured by:

$$Error\ Rate = \frac{Number\ of\ invalid\ character\ pairs}{Total\ number\ of\ extracted\ pairs} \times 100 \tag{4.1}$$

Note that the number of characters and ligatures differs between these methods, because the different approaches solve the problem of layout variation of the OCR and the transcription. The first part of the experiments shows that aligning the OCR line lattice with small text line segment is recommended for solving the problem of OCR segmentation. Figure 4.9 shows the coverage of the text output of each alignment approach regarding the original ground truth based on the text length for 106 documents.



(a) Small Text Segment



(b) Page-Wise

Figure 4.9: Length of the alignments output for each page by applying different alignment approaches compared to the original transcription, (a) Small Text Segment, (b) Page-Wise. The pages number is represented on the horizontal axis. On the vertical axis, the length of the pages is shown based on the number of characters, and spaces that exist in the original ground truth (blue color) and the number of the characters that are produced by the implemented method in (a), and (b), in red color.

In the second part of the experiment, the approaches are applied on a transcription edition which is different from the original documents to show their performance under text modification, and layout variations. The transcription is used from project Gutenberg-DE. The imperfect transcription was 90.53% correct compared to the original transcription. It has a lot of missing text and many words which are written in different way from the original documents but have the same meaning, such as: "*Kantonversammlungen*" instead of "*Caton-Versammlung*", "*Dreißigjährigen*" instead of "*30jährigen*", "*Gesamtökonomie*" instead of "*Gesammt-Oekonomie*", "*Teilen*" inst of "*Theilen*", and "*fünf Uhr*" instead of "*5Uhr*". The transcription has no hyphenated words.

A hyphenated WFST is also designed to hyphenate the transcription text so a match is made in the original text documents. The transcription edition has no line, page breaks information, extra or missing text, and no hyphenation. It has different vocabulary which has the same meaning, i.e., in the original document the editor used "zugleich", and in other transcription "ebenso" is used. Capitalization or small letters are different from the original documents.

The number of the returned characters and ligatures and the error rates are shown in Table 4.4. Figure 4.10 shows the coverage of the text output of each alignment approach

Table 4.3: Performance evaluation results of the approaches applied on imperfect transcription edition.

| Alignment Approaches | Total Number of obtained Characters | Error Rate |
|---|---|---|
| Small Text Segments with Hyphens | 121,737 | 2.0% |
| Small Text Segments No Hyphens | 121,902 | 2.33% |
| Book-Wise with Hyphens | 127,022 | 2.54% |
| Book-Wise No Hyphens | 109,702 | 3.87% |

regarding the original ground truth based on the text length.

The small text segments approach is applied by using hyphenation on the words in the text segments. There is no information available about which word is at the end of the line and if they are hyphenated or not. Therefore, the WFST-based approach is designed so that it can skip the hyphen transition from transcription if it does not exist in the OCR lattices. The model is also capable of producing the hyphenated parts which exist in the OCR lattices and not in the transcription. The alignment approach which used the hyphenation WFST-based shows better performance in aligning the hyphenated OCR's words with the auto-hyphenated transcription words, such as "*Pas-*" in "*Passagierboote*" the model aligned the lattices part "*s*" which the transcription "*s*", and the lattice part "-" with the transcription "-" respectively. While applying the approach without hyphenation failed in several cases and aligned lattice part "*s-*" with the "*s*" in the transcription. Figure 4.11 shows the output of the aligned pairs of some ligatures. Those aligned pairs will be used for training purposes.

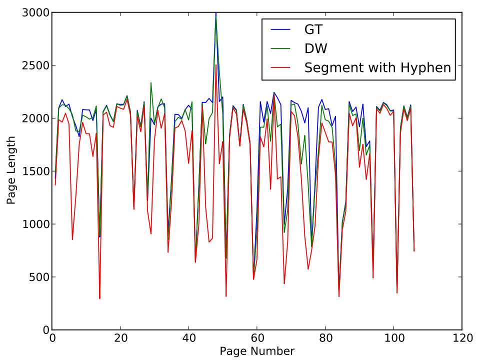
The command line is $transcription - alignment[options] * .gt.txt$. The options are $-p$ is aligning with page-wise model, $-s$ is aligning with small text segment model and $-b$ is aligning with book-wise model. For each alignment approach, information is provided to know how well the line aligned, and if there is over segmentation. The alignment provides text lines an output and also an isolated character database where the the aligned pairs are stored. The implemented approaches are simple and an editing tool is provided which allow for easy viewing and editing of the aligned pairs, as shown in Figure 4.12. The raw output of the alignment is also shown with a message showing that the lines are correctly aligned to the right transcription.

### 4.4.2 Experimental Results of Dehyphenation of Noisy Transcription

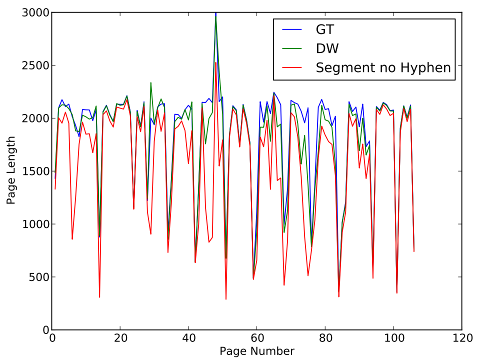
This section describes the datasets and the obtained transcriptions which are used in the experiments. The setup of the experiments is described.

#### OCR and Datasets

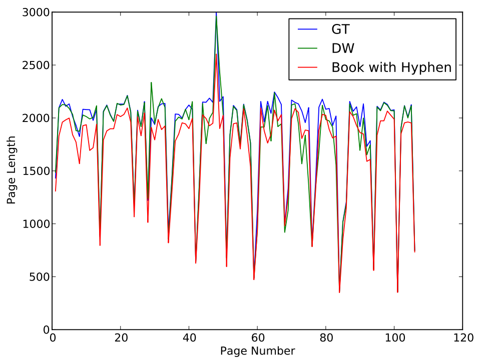
The datasets are generated from the raw output of the OCRopus system after applying it on English UWIII dataset and German Fraktur Documents obtained from "Wanderungen durch



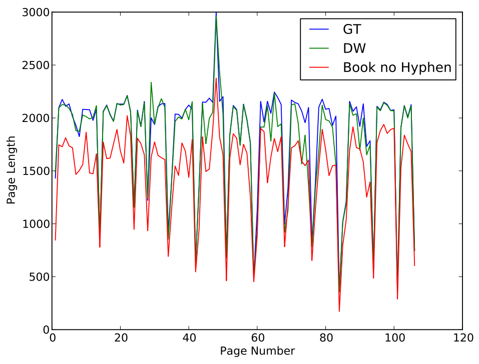
(a)



(b)



(c)



(d)

Figure 4.10: Length of the alignment output for each page by applying different alignment approaches compared to the original and different edition (Downloaded) transcription, (a) Small Text Segment with Hyphenation, (b) Small Text Segment without Hyphenation (c) Book-Wise with Hyphenation, and (d) Book-Wise without Hyphenation. The pages number is represented on the horizontal axis. On the vertical axis, the length of the pages is shown based on the number of characters that exist in the original ground truth (blue color), the downloaded transcription with different line breaks and contents (green color), and the number of the characters that are produced by the implemented models in (a), (b), (c), and (d) in red color.

die Mark Brandenburg" volumes 1862-1889 by Heinrich Theodor Fontane. The purpose is to learn the tokens which appear with or without a hyphen, with the aim of finding out whether the hyphen is related to line breaks, due to combing words or misrecognition of a symbol as a hyphen by the OCR. Therefore, it is important that the text contains words with hyphens in different positions based on the number of syllables. The ground truth is represented in words and corresponding labels that refer to whether the word has a hyphen or not.

In the first part of the experiments, the approaches are applied on the Fraktur documents. Three volumes are chosen for training and one volume of a different nature for obtaining the

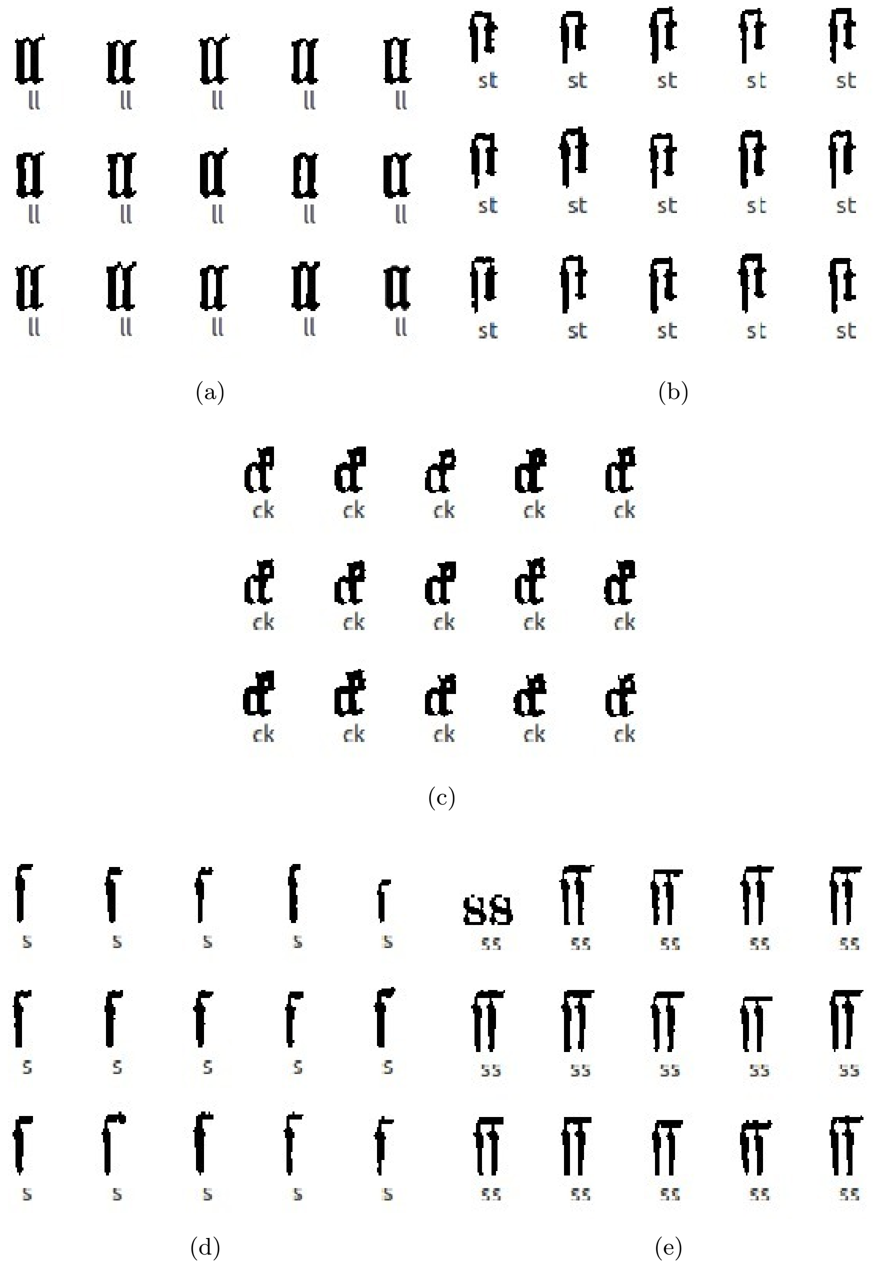
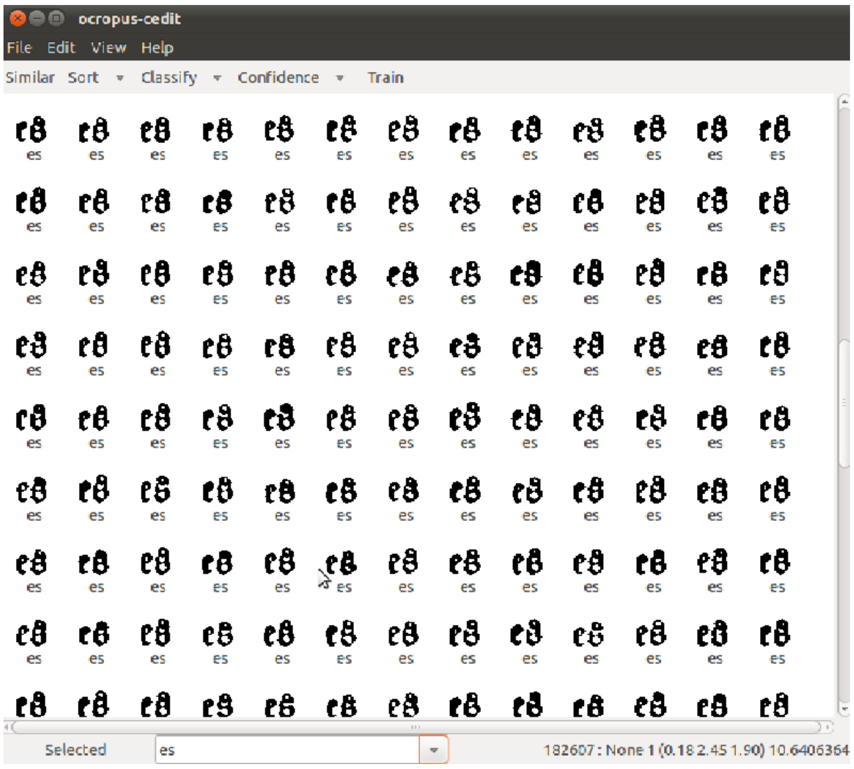


Figure 4.11: Output of the alignment tuple of the Classes: (a) "*ll*", (b) "*st*", (c) "*ck*", (d) "*longs*", and (e) "*ss*".



182607 : None 1 (0.18 2.45 1.90) 10.6406364

Figure 4.12: Editing and output of the alignment tuple of the Class: "*es*".

Table 4.4: Performance evaluation results of the dehyphenation classifiers applied on cleaned data and noisy OCR's Results. The experiments are done first by classifying the hyphens of a clean transcription, then the classifiers are applied on a noise transcription. English script from the UWIII dataset, and German Fraktur from Fontane documents are used.

| Test Sets | Regular Expression | Naive Bayes | Decision Trees | Maxent |
|---|---|---|---|---|
| Cleaned English | 60% | 97% | 99% | 98% |
| Noisy English | 64% | 97% | 98% | 97% |
| Cleaned Fraktur | 76% | 97% | 98% | 97% |
| Noisy Fraktur | 74% | 96% | 97% | 96% |

OCR of those documents and using them for testing. The models are trained on 11,239 combinations of hyphenated and compound patterns. In order to evaluate the implemented models, a portion of 1,248 tokens is reserved for testing. Then, the models are evaluated on 2,425 tokens which are extracted from the OCR output. The results appear in Table 4.4. The table shows the performance of the three classifiers: Decision Trees, Naive Bayes, and Maximum Entropy classifiers (Maxent) using the NLTK toolkit 2.0 in Python. All classifiers are implemented in Python and applied using the NLTK toolkit 2.0 [LB02] under Linux. Running times for the classification is fast on a modern desktop PC with four cores and 8 GB RAM. The baseline method is the Regular Expression-based approach of [GT94].

In the second part of the experiments, the approaches are applied on the English documents from the UWIII datasets. The extracted combination of hyphenated and compound patterns are split in a training set of 6,262 and a test set 932. The models are evaluated on 2,623 tokens which are extracted from OCR's output. The results in Table 4.4 shows the performance of the three classifiers on the test sets.

The approaches process words with Unicode, letters with diacritics, capitalization, digits, and compound words which are not processed in [GT94]. For example, "Flüß-`\n` chests Flüßchests", "verhältniß-`\n` mäßig verhältnißmäßig", "treff-`\n` lichzu trefflichzu", "mehr-`\n` genannten mehrgenannten", "Schermtzel-See", and "Pfälzer-Dörfern".

The compound words get a hyphen of a line break, if it appears at the end of the line "Re-`\n` lief-Figur Relief-Figur". The approaches classify the words even if they have spell errors "Werbelliner-Forstj". They also classify compound words that start with small letters "neuknigliches" or capital letters "Stammes-Eigenthmlichkeiten,", and "MORITZ-KIRCHE". The approaches work with combined words with digits under recognition errors"16-Jahrhuntertg". They also classify long compound words such as "General-Feldmarschall-Lieutenant,".

The classifiers misclassified the compound word "grell-echg" and removed the hyphen. The second token "echg" is originally the word "echt" and it has a recognition error. Due to this error the second token did not have a tag and it starts with small letter, therefore it was misclassified. The compound German words usually start with capital letter. The compound word "Schlo-geseffenen" is also misclassified, which is originally "Schlo-Geseffen". Those examples were confused cases for the classifiers.

The Naive Bayes misclassified "Berechtig-`\n` es Berechtiges" and classified it as a compound word because "es" is a personal pronoun and it has a tag. The word "Cister-`\n` zienser-OrdenC Cisterzienser-OrdenC" is misclassified. The word "bedenklich-`\n` pittoresken bedenklichpittoresken" is misclassified as a compound word did not remove the hyphen because both tokens have a tag and they are words.

## 4.5 Discussion

The section includes two parts. Section 4.5.1 describes the outcomes of the alignment approaches. Section 4.5.2 shows the discussion of the dehyphenation approaches.

### 4.5.1 Discussion of Alignment Approaches

The implemented Small Text Segment model aligns with or without the hyphenation model and also produces correct aligned pairs for training purposes even if the transcription is imperfect and has many differences from the original documents. The model is applied on a transcription which has missing text contents, different words, terms, line breaks, capital, and small letters. The WFST-based alignment approaches are simple and give the best matches easily. It allows us to add rules which help to skip mismatches or correct OCR characters in the lattices. The book-wise approach is used because it is suitable when no page beak information is available and the line order is different, especially when the transcriptions have a lot of mismatches in the contents. The approach is directly designed for WFST to have a faster and direct alignment between the OCR lattices and the transcription.

The first part of the experiments shows that aligning the OCR line lattice with small text lines segment is recommended for solving the problem of OCR segmentation. In the second part of the experiments, the performance of the alignment approaches are evaluated when the original transcription does not exist, and the only available transcription is an imperfect transcription with many errors instead, e.g., text is having different line breaks, different words, and sentences and many differences in spelling and vocabulary. In some pages 50% of the text contents missing. The transcription has no hyphenations. The alignment approaches are evaluated with and without the hyphenation model. It shows that the alignment approaches still can produce correct aligned pairs even if it do not have any information about the hyphens. It just produces those characters which are touched with the hyphen as “*r*-” instead of “*r*” and “-” when the hyphenation model is not used. An application could report those pages and lines which have a bad alignment result, so that the user can edit their transcriptions separately. The generated aligned pairs are stored in an isolated character database with few errors and can easily correct the pairs and are assigned the correct label to the character segment.

The straight-forward approach would fail in the case of a noisy transcription, because no unique words would be found, as in Manmatha et al. [YM11]. This issue is solved in the implemented WFST-based approach. Noteworthy, the WFST-based approach is still easy to be implemented. In the case of a transcription with many errors, aligning the line against the whole transcription can be an option to avoid the differences in the transcription, and produce correct pairs, but also using extracted text segment with adaptive length can provide good results faster. The imperfect transcription has some extra text lines and paragraphs between the desired text lines. The number of the text lines can be increased to enlarge the segment of the second approach when the documents are aligned with the imperfect transcription, so that the range of the segment is expanded and avoids aligning only with mis-correspondence lines. A punctuation model can be easily added to the WFST as in Ref. FIFB11, to avoid the different kind of punctuations which is used in the imperfect transcription, in some cases such as “//” which is replaced by the “ $\gg$”. The advantage of implemented approaches is that they are be applied to documents of other scripts easily and accurately.

### 4.5.2 Discussion of Dehyphenation Approaches

In Section 4.3.3, a machine-learning based classification approach is implemented for the dehyphenation of OCR outputs.

With this approach the following issues are solved: supporting the Unicode, can be used for different languages, and solving the problem of processing hyphenation with combined words and digits which are not handled with the other approach. The approaches are robust against OCR's errors which include spell errors and substitution of symbols, i.e., dashes with hyphens due the misrecognition. Particular features of language data are used for classifying types of hyphen in OCR's results such as: word length, length of the hyphenated tokens, capitalization, and part-of-speech tagging which have not been used before. It was found that these processes could handle several various cases of hyphenations and return significantly better results. No OCR correction is applied before the dehyphenation and thus it was found again that these approaches could handle various cases of hyphenation.

An appropriate feature is the length of both tokens joined by a hyphen, e.g., the length of the words is longer than the length of the hyphenated tokens. Capitalization is one of the most appropriate features because the second part of the compound word could start with capital which would make it distinguishable compared to the second part of hyphenated word at the next line. However, the compound words in English start with lowercase. Also some compound words have short length of words. Therefore using part-of-speech tagging support to distinguish the real words from the hyphenated parts or part of compound words. The hyphenation of a compound word at the end of the line, might rarely happen, but the two parts of the hyphenated word could have a tag by using the tagger for German, and the hyphen is kept for the words combination. To build good models, the OCR results need to be observed to select suitable features which can provide a good separation of the data. The approaches are applied on OCR results and can be applied on any text corpus. The classifiers perform well on large dataset and classify them in short time. They are simple to understand and interpret and can handle both numerical and categorical data.

# Part II

# Language Modeling using Long Short-Term Memory (LSTM)

# Chapter 5

# Normalizing Historical Orthography

Historical text presents numerous challenges for contemporary different techniques, e.g., information retrieval, Optical Character Recognition (OCR), and Part-of-Speech (POS) tagging. In particular, the absence of consistent orthographic conventions in historical texts presents difficulties for any system which requires reference to a fixed lexicon accessed by orthographic form. For example, a language modeling or retrieval engine for historical text produced by OCR systems, where the spelling of words often differ in various way, suffer from disambiguities. It is important to aid those techniques with the rules for automatic mapping of historical wordforms.

In this chapter, a new technique is implemented to model the target modern language by means of a recurrent neural network with long-short term memory architecture. Because the network is recurrent, the considered context is not limited to a fixed size, especially due to memory cells which are designed to deal with long-term dependencies.

The contribution[1] of this chapter is as following:

1. Novel alignment techniques are implemented to prepare the training pairs in historical and modern forms. The techniques are called Character-Epsilon Alignment and Size Normalization with Epsilon.

2. The implemented alignment techniques translate the orthographic rules to edit operations and allow insertion, deletion, and substitution operations.

3. The designed approach is applied on historical texts with the absence of consistent orthographic conventions.

4. Approaches are applied to transform wordforms from Early New High German (ENHG) 14th - 16th centuries, as shown in Figure 5.1.

5. The novel implemented alignment approach using LSTM outperforms the state-of-the-art approaches designed by Bollmann et al. [BPD11a, BPD11c].

[1] This chapter is based on the Al Azawi work in [AAAB13].

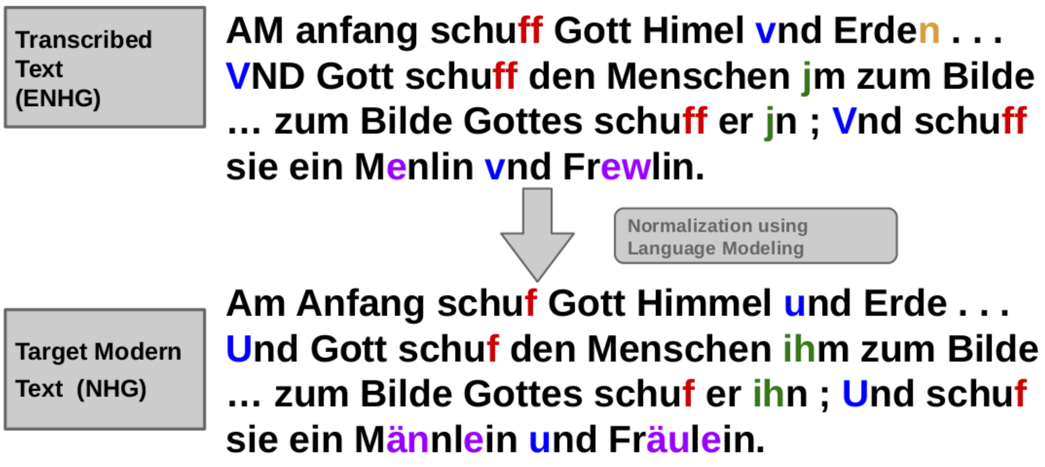

Transcribed Text (ENHG)
AM anfang schuff Gott Himel vnd Erden . . . VND Gott schuff den Menschen jm zum Bilde … zum Bilde Gottes schuff er jn ; Vnd schuff sie ein Menlin vnd Frewlin.
Normalization using Language Modeling
Target Modern Text (NHG)
Am Anfang schuf Gott Himmel und Erde . . . Und Gott schuf den Menschen ihm zum Bilde … zum Bilde Gottes schuf er ihn ; Und schuf sie ein Männlein und Fräulein.

Figure 5.1: Example of the original ENHG and the modernized New High German (NHG) which is used as a target to be transformed from the original form using the implemented normalization approach (LSTM).

6. The evaluation shows the accuracy of the implemented model on the known wordforms is 93.90% and on the unknown wordforms is 87.95%, while the accuracy of the existing state-of-the-art combined approach of the wordlist-based and rule-based normalization models is 92.93% for known and 76.88% for unknown tokens.

7. The implemented LSTM approach predicts unknown orthographic rules which were not seen during the training phase, while the rule-based and wordlist are unable to normalize unseen tokens in their training rules or words list.

The chapter is organized as follows. In Section 5.1.1 and 5.1.2, two state-of-the-art methods are described which are wordlist-based and rule-based. In Section 5.2, the novel implemented method is described. Section 5.3 shows the experimental results. Finally, Section 5.4 presents the conclusions.

## 5.1 State-of-the-Art Approaches

In this Section, two state-of-the-art approaches will be described. In Section 5.1.1 the Wordlist-based approach is described. The Rule-based approach is shown in Section 5.1.2.

### 5.1.1 Wordlist-based Approach

This approach consists of wordlists that map historical wordforms to their modern wordforms. The wordlist is generated by aligning sentences and words between two languages. It is important to specify that the two languages are the source language and the target language in

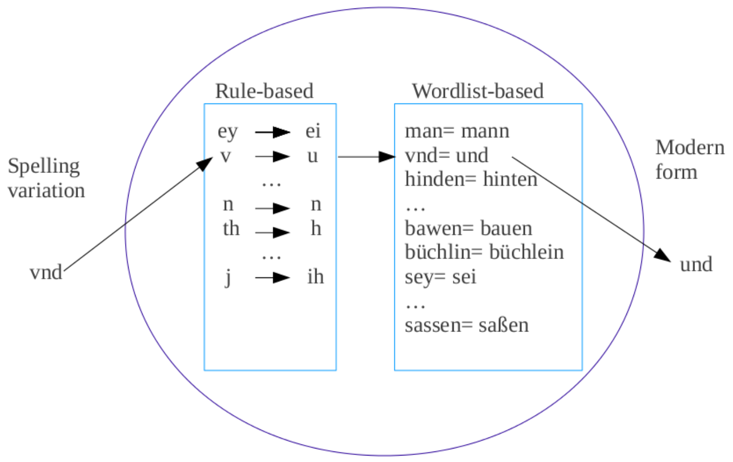


Figure 5.2: Combined approach of Rules-based and Wordlist-based approach. The Figure shows samples of rules from the mapping of historical to modern wordforms.

order to correctly apply the word alignment. Then, a pre-processing is applied to clean up the corpora of the two languages, set every word in lower case, and separate every word from each other. The word alignment can be one-to-one, Null-to-one, and many-to-one. The alignment is doing using HMM model. Then, the alignment output need to be checked and corrected, if it is needed. The alignment is done using the Gargantua [BF10] and the GIZA toolkit [ON03]. Running this technique requires a lot of memory. The historical strings are matched to the modern strings, then the tool finds the matched tokens and outputs the parallelized text. The alignment of the parallelized historical and modern text is done automatically, but it is time consuming, and labor-intensive because specialists (historical linguists) are needed to check the aligned generated pairs. The modern wordforms are generated by aligning the historical words with n-gram model and substituting an old wordform with its modern counterpart. The n-grams are widely used in statistical language model for speech recognition, machine translation, OCR, and normalization of historical text [BPD11c, Moh03b, LNCPCA10].

### 5.1.2 Rules-based Approach

The rules are derived from aligning the historical and modern words and are weighted according to their frequency. The rules are obtained by applying the edit distance algorithm to extract single characters to identity rules and context-based confusions between the historical and modern wordforms as replacement rules. Those replacement rules are used to build an approach which can solve the problem of the historical orthography where the wordforms have evolved over time and have many different variants.

The rules are represented in insertion, deletion, and substitution operations. Whenever multiple rules are applicable at the same position within a word, the rules that have a high rank

are applied [BPD11c]. The identity rule is represented by mapping the character to itself, for example, $n \rightarrow n$. The substitution rule means to map a character or multiple characters to another one, for example, $v \rightarrow n$, $j \rightarrow ih$. The insertion rule is used for inserting a character $\epsilon \rightarrow l$. The deletion rule is used for delete a character $f \rightarrow \epsilon$.
The rule-based approach using the Levenshtein edit distance is used in speech recognition [Moh03b] and OCR [LNCPCA10] to assist the language model in the correct recognition results.
Figure 5.2 shows a combined approach based on both rules- and wordlist-based approaches. The samples in the Figure represent the rules, historical and modern wordforms, e.g., "vnd" represents an input word in historical form which is passed through the rules-based and then wordlist-based approach, and transformed to modern wordform "und". However, historical wordforms which have not existed in the wordlist are only passed through the rule-based approach.

## 5.2 Designed LSTM Approach

A new technique based on LSTM recurrent networks is implemented to solve the problem of transforming historical wordforms into modern wordforms and vice versa which had very little attention in the past. An introduction for LSTM is provided in Section A.2.3.

For training, the historical wordforms are used with their corresponding modern wordforms. The training pairs are prepared using two alignment techniques, as described in Section 5.2.1 and Section 5.2.2 which are considered a pre-processing step for feature extraction. Two different techniques are applied to align the training pairs, the first uses the Levenshtein edit distance and the second uses an empirical number of epsilons. The alignment technique of two strings finds the similarities and differences between them and can be interpreted as point mutations. If they share common characters, mismatches, and gaps then insertion or deletion mutations are introduced in one or both lineages at the time since they diverged from one another. The training and testing pipeline is shown in Figure 5.3.

### 5.2.1 Designed Character-Epsilon Alignment

The first method Character-Epsilon alignment uses the Levenshtein alignment technique [Lev66a] to align two strings $A$ and $B$ and find an optimal alignment to give a better score. To compute an alignment that actually gives the score, it starts from the bottom right cell, and compare the value with the possible sources (Match, Substitution, Insertion, and Deletion) to see which it came from. In the case of Delete, $A_i$ is aligned with a gap and for Insert, $B_j$ is aligned with a gap. Otherwise $A_i$ and $B_j$ are aligned to accomplish this goal.

First the epsilon after each character of the historical wordform is inserted, then alignment is applied using the Levenshtein edit distance between the historical and modern wordforms to

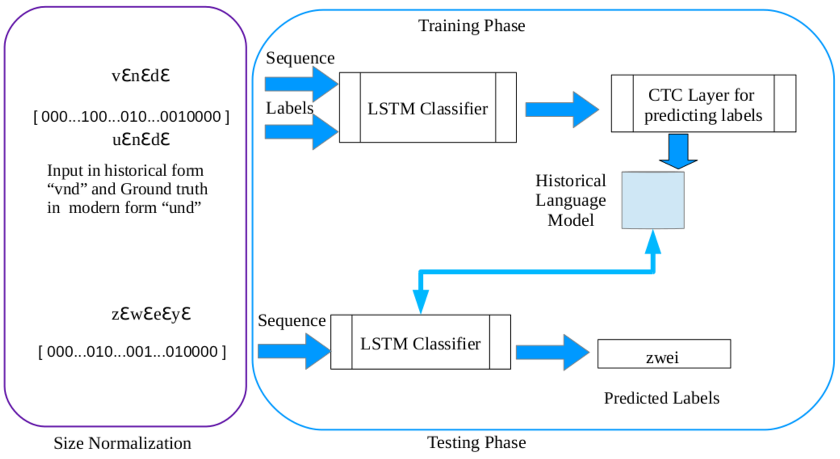


Figure 5.3: Training and Testing Pipeline. Historical wordform in text format is converted to 1D vector and it fed to the LSTM network along with its ground truth (modern form). CTC layer performs the output transcription alignment. Finally the network is trained and is used as historical language model.

obtain the optimal aligned character pairs, for example:

1. Input historical form: jrem
2. Adding epsilon to the historical wordform: j $\epsilon$ r $\epsilon$ e $\epsilon$ m $\epsilon$
3. Applying alignment to obtain:
   historical wordform = j $\epsilon$ r $\epsilon$ e $\epsilon$ m $\epsilon$
   modern wordform = i h r $\epsilon$ e $\epsilon$ m $\epsilon$

Another example where insertion is required:

1. Input historical form: jr
2. Adding epsilon to the historical wordform: j $\epsilon$ r $\epsilon$
3. Applying alignment to obtain:
   historical wordform = j $\epsilon$ r $\epsilon$
   modern wordform = i h r $\epsilon$

Another example where substitution and deletion are required:
historical wordform = c $\epsilon$ r $\epsilon$ e $\epsilon$ u $\epsilon$ t $\epsilon$ z $\epsilon$
modern wordform = k $\epsilon$ r $\epsilon$ e $\epsilon$ u $\epsilon$ $\epsilon$ $\epsilon$ z $\epsilon$

Then, the LSTM network is provided with feature vector of the historical wordforms and the corresponding groundtruth labels in modern wordforms for training purposes. For testing purposes, the LSTM trained model is applied after adding epsilon $\epsilon$ behind each character in the historical wordform of the test token, and then predicting its modern wordform (see Section 5.2.3.

### 5.2.2 Designed Size Normalization with Epsilon

The second type of alignment is Size Normalization with Epsilon and it is applied by using an empirical number of epsilons to assist the alignment of the characters of historical and modern wordforms. Wordform size normalization is an essential step in the feature extraction of the tokens and it is started by adding three epsilons to small size historical or modern tokens. Then, the updated historical and modern wordforms are aligned to have the optimal transformation operations (deletion, insertion, substitution) between corresponding characters. The addition of epsilons is not employed for equal size historical and modern wordforms.

1. Input historical wordform = j m
2. Appending epsilon at the end of the string = j m $\epsilon$ $\epsilon$ $\epsilon$
3. Applying alignment to obtain:
   historical wordform = j m $\epsilon$ $\epsilon$ $\epsilon$
   modern wordform = i h m $\epsilon$ $\epsilon$

The two different size normalization and alignment on the historical and modern wordforms are applied before LSTM training. It is applied on the normalization from historical to modern forms and vise versa.

### 5.2.3 String Encoding

For encoding a string, a sequence of vectors is used where each vector has a length corresponding to the size of 300. Each character in the string is mapped to its code point which is used in the binary feature representation, as shown in Figure 5.3.
Figure 5.3 shows the complete training and testing pipeline and the normalized historical wordforms along with their transcriptions are fed into the network, which perform the forward propagation step first. Alignment of output with associated transcriptions is done in the next step and then finally the backpropagation step was performed.
After each epoch, training and validation errors are computed and the best results saved. When there was no significant change in validation errors for a pre-set number of epochs, the training was stopped. There are three parameters, which need to be tuned; namely input-wordform size, hidden-states size, and learning rate. It will be explained in detail in Section 5.3.

### 5.2.4 LSTM Networks Configuration

The configuration of the network and the number of weights mapping between and within layers is shown in A.2.3. Training of the network proceeds by choosing text input lines randomly from

Table 5.1: Character accuracy on Luther Bible Testsets using LSTM, Rule-based, Wordlist-based approaches, and the combined approach of Rule-based and Wordlist-based.

| Testsets | Rule-based | Wordlist-based | Combined | LSTM |
|---|---|---|---|---|
| All | 91.00% | 91.98% | 92.93% | 93.90% |
| Unknown | 76.88% | 40.88% | 76.88% | 87.95% |

the training set, performing a forward propagation step through the LSTM and output networks, then performing forward-backward alignment of the output with the ground-truth, and finally performing backward propagation.

## 5.3 Experiments

The experiments are described in this section. Section 5.3.1 explains how the Luther Bible dataset was extracted. In Section 5.3.2, the experimental results are shown.

### 5.3.1 Luther Bible Dataset

The dataset is state-of-the-art established by Bollmann et al. [BPD11c]. In the experiments 219,948 historical wordforms are used with their corresponding modern wordforms for training purposes. Testing was done on 109,973 historical wordforms and their corresponding modern wordforms in UTF-8 encoded text format which are used as ground truth to evaluate the generated modern wordforms by the implemented normalized approaches. The dataset is generated from OCRed historical text of the Martin Luther bible 1545 and the revised modern version.

### 5.3.2 Experimental Results

Table 5.1 shows the evaluation of the approaches using the whole testset and a separate evaluation was done on word pairs that were not seen during the training denoted as "unknown". The unknown wordform is a subset from the testset. The evaluation shows an effective performance of the LSTM on the whole testset 93.90%. LSTM shows significant results 87.95% on the unknown testset compared to other approaches.

The LSTM is trained on the features of the first preprocessing approach 5.2.1 by adding epsilon after each character in the historical form and then aligning with the modern form, which gives better results. For testing, epsilon is added after each character in test historical form, and then the modern form is predicted using the trained LSTM network. While using addition of three epsilons at the end of the wordform, as other preprocessing step 5.2.2 for the LSTM, delivered 86.37% for unknown testset.

The implemented LSTM approach is compared with the combined model of rule- and wordlist-based approaches. The combined model shows lower performance than the LSTM,

Table 5.2: Top confusions of the character-epsilon alignment with LSTM.

| Operations | Confusions |
|---|---|
| Substitution | ohl : oll |
| Substitution | as : aß |
| Insertion | $\epsilon$ : s |
| Deletion | f : $\epsilon$ |
| Deletion | h : $\epsilon$ |

Table 5.3: Top confusions of the adding three epsilons alignment with LSTM.

| Operations | Confusions |
|---|---|
| Substitution | a : h |
| Substitution | oho : oll |
| Substitution | uh : un |
| Substitution | ehe : eue |
| Insertion | $\epsilon$ : n |
| Insertion | $\epsilon$ : s |
| Insertion | $\epsilon$ : e |
| Deletion | t : $\epsilon$ |
| Deletion | f : $\epsilon$ |
| Deletion | d : $\epsilon$ |
| Deletion | e : $\epsilon$ |

but performs better than using the rule-based or wordlist-based individually each one as a separate approach. The wordlist-approach is unable to generate a modern wordform for unknown tokens. Furthermore for the rules mismatches may link a historical spelling variation to a wrong modern word. LSTM is well suited to work with learning sequences and achieved the best known results in unsegmented connected handwriting recognition.

The LSTM networks are trained with a hidden-state of different size 40, 60, 80, 100, 120, 140, and 200. The optimal error rates are obtained when the size of the hidden-states is in a range between 120 - 200. It takes 782 - 1097 minutes for training. When the size of the hidden-states is in a range between 40 - 100, it takes 251 - 594 minutes for training respectively. The time increases linearly with an increase in the number of hidden-states. After the model has been trained, the predictions are fast. The accuracy is measured by using string Levenshtein edit distance. The most appropriate number of hidden-states were determined keeping the learning rate constant at 0.0001.

Most of the confusions which occurred while predicting the modern form using the character-epsilon alignment are very few and the error types are shown in Table 5.2 such as *ohl* confused with *oll*. The second alignment used adding the three epsilons at the end of the word and the top confusions are presented in Table 5.3.

The transformation of the orthographical variants differs from word to word. And it is also

Table 5.4: Output of LSTM on predicting the modern form.

| Prediction | Historical form | Modern form | Rule |
|---|---|---|---|
| und | vnd | und | v $\rightarrow$ u |
| über | vber | über | v $\rightarrow$ *ü* |
| erwählet | erwelet | erwählet | e $\rightarrow$ *ä* h |
| egyptenland | ägyptenland | egyptenland | e $\rightarrow$ *ä* |
| ihr | jr | ihr | j $\rightarrow$ i |
| ich | jch | ich | j $\rightarrow$ ih |
| verführet | verfüret | verführet | h $\rightarrow$ $\epsilon$ |
| sohn | son | sohn | $\epsilon$ $\rightarrow$ h |
| kalzedonier | calcedonier | chalzedonier | $\epsilon$ $\rightarrow$ h, c $\rightarrow$ z |

Table 5.5: Output of LSTM on predicting the modern form using the preprocessing with Size Normalization with Epsilon.

| Prediction | Modern form | Historical form |
|---|---|---|
| zwelff | zwölf | zwelff |
| das | daß | das |
| euer | eure | ewr |

possible that one character is changed to many characters and vice versa. The LSTM shows the ability predict the modern form correctly even if there are more than one transformed spelling variants in single word form. LSTM outperforms various transformation variants. It is also capable of predicting the character when it is transformed into various modern combinations in different historical word form,as shown in Table 5.4. It is also capable of deletion or insertion of characters such as $h \rightarrow \epsilon$ or $\epsilon \rightarrow h$.

The trained LSTM using addition of three epsilons at the end of the wordform performed good on the rules of change "v" to "u", and "j" to "i" and some others. But, it produced errors which was not given in the previous LSTM, as shown in Table 5.5.

The LSTM is trained to perform transformation of the modern wordforms to their historical wordforms. Different preprocessing strategies are applied, first approach by insertion of epsilon after each character which is called LSTM1. The other approach is adding three epsilon at the end of the token which is called LSTM2. The two approaches are compared using the LSTM with the baseline. The results is shown in Table 5.8. The baseline approach means that the tokens are passed with no changing. LSTM 1 performs very well in change the "u" to "v", the "e" to "ä", insertions of "d" or "t", but LSTM 2 could not fulfill the all orthographic rules, as in Table 5.6: In the examples of Table 5.7 both LSTMs perform well.

Table 5.6: Output of LSTM on predicting the historical form using the preprocessing with Character-Epsilon Alignment.

| Prediction | Historical form | Modern form |
|---|---|---|
| vnd | vnd | und |
| todte | todte | tote |
| kurtzes | kurtzes | kurzes |

Table 5.7: Output of LSTM on predicting the historical form using the preprocessing with Character-Epsilon Alignment and Size Normalization with Epsilon.

| Prediction | Historical form | Modern form |
|---|---|---|
| nehe | nehe | nähe |
| vberkeme | vberkeme | überkäme |

## 5.4 Discussion

The results presented in this Chapter show that the LSTM yields excellent normalization results for historical orthography. There are several indications that the LSTM-based approach generalizes much better to unseen samples than previous approaches applied to historical spelling variants.

Previously, the artificial neural network was not used in historical orthography tasks. The CTC layer is responsible in finding and aligning with the corresponding labels. The CTC is used for such purposes for hard tasks such as handwritten recognition. Another indication is the excellent performance of the LSTM-based approach. It is a simple approach and needs less time and effort to train. The prediction using LSTM is fast. However, other approaches are time consuming, labor-intensive and require effort to construct and combine different components.

The rule-based approach might not be able to cover all the spelling variants when the OCR does misrecognition of characters. The misrecogniton leads to a mismatch of a historical spelling variation to a wrong modern word. Some approaches pass unseen samples unchanged when they are not in their model. Wordlist approach is unable to process unseen samples. The case of synonyms, if they exist in the training data, the model is able to learn it, and do the appropriate normalization in the given context. However, all the tokens are processed in the testsets through the implemented LSTM, and the model is able to predict quickly all different spell variants significantly.

Table 5.8: Character accuracy on Luther Bible Testsets using different LSTM Networks and baseline approaches.

| Testsets | baseline | LSTM 2 | LSTM 1 |
|---|---|---|---|
| Seen Tokens | 89.73% | 91.94% | 93.40% |
| Unknown Tokens | 87.16% | 87.96% | 89.17% |

# Chapter 6

# Learning-Based Character-Level Corrections in Multi-Script Recognition Systems

Handwritten and printed text recognition research focuses more and more on challenging scripts and bad quality images, as these are difficult recognition tasks. Language modeling techniques are required to improve the recognition results. While dictionaries are built with finite vocabularies, however, a language model should be capable of efficiently creating infinite dictionary corrections. Therefore, a fast and accurate technique with a capability of predicting unknown tokens is needed.

In this Chapter, a new method is introduced to normalize the strings and correct the OCR errors, using the Recurrent Neural Networks (RNN) with Long-Short Term Memory (LSTM). A second method is the error model using Weighted Finite-State Transducers (WFSTs) with context-dependent confusion rules in Chapter 3 is used for the first time for difficult to recognize Urdu scripts. Both methods are applied on OCR results of Latin and Urdu Script. Urdu script is challenging for OCR systems.

For building an error model using context-dependent confusion rules, the OCR confusions which appear in the recognition outputs are translated into edit operations using Levenshtein edit distance algorithm. The newly designed LSTM model avoids the calculations that occur in searching the language model and makes the language model capable to correct unseen incorrect words. The implemented generic approaches are language independent.

Building and evaluating a post-processing system for OCR corrections are discussed. Two different language models are used: error model transducer in the form of WFST and Character-Level correction in LSTM. The contributions are[1]:

C1: The error model is built using context-dependent confusion rules and WFSTs. It is a new

[1] This chapter is based on the Al Azawi work in [AAUHLB14].

contribution for Arabic and Urdu script.

- Based on the confusion matrix that the OCR produces and is dependent on the context of the strings (OCR results).
- Designed error model in Chapter 3 returns a lower error rate than the state-of-the-art single character rule-based approach [HNH08].
- Error model is extended to more challenging script such as Urdu Nastaleeq (Arabic) by extracting new rules in this Chapter.
- Tested by implementing a WFST from the Levenshtein edit distance relations.
- Context is a new idea instead of single character rules. This helps to fit the confusion rule in the proper string where it belongs and brings the string to the corrections.
- Language model can be as simple as a list of finite words compiled into WFSTs.
- Frequencies of a token or a rule in the corpus are converted to weights in the WFSTs.
- Rule that is ranked higher is applied, if multiple rules are applicable at the same position within a word in the error model approach.
- Rule-based fails when the cost of appropriate correct candidate is higher than the cost of other correct candidates for the same input token.

C2: The LSTM based approach.

- A powerful method trained to learn corrections of recognition errors.
- Designed specifically to overcome limitations of RNN.
- Ability to remember the target association between irrelevant input and target events even for very long time lags [GLF+09].
- A new Character-Level alignment is designed to normalize the string's length before the training of the LSTM, as described in Section 6.3.1.
- Language independent approach.
- Normalization of the historical orthography for OCR historical documents.
- Used in Latin and Urdu script.
- Recognition and language models results examples of Latin and Urdu Nastaleeq are shown in Figure 6.4.
- Experiment results show that the designed LSTM model has better performance than the error model on the known and unknown data which unseen during the training.
- The supervised LSTM model is compared with the context-dependent error model and state-of-the-art single character rule-based methods. The evaluation on Latin script shows an error rate of LSTM is 0.48%, error model is 0.68%, and the rule-based

model is 1.0%. The evaluation shows the accuracy of the LSTM model on the Urdu testset is 1.58%, while the accuracy of the error model is 3.8%, and OCR recognition result is 6.9% for Urdu testset.

- Best performance is obtained by applying LSTM in both Latin and Urdu script. As such, experiments show that LSTM performs well in language techniques, especially, post-processing.

The chapter is organized as follows. Section 6.1 describes the state-of-the-art single character rule-based model. In Section 6.2 and 6.3, the context-dependent error model and LSTM networks are described. Section 6.4 contains the experimental results. Finally, Section 6.5 presents the conclusions.

## 6.1 Single Character Rule-Based Approach

The single character rule-based approach is widely used in various applications such as spell-checkers and handwriting recognition. Pirinen et al. [PH12] and Hassan et al. [HNH08] proposed an error model to aid the language model in a spell-checker to correct misspelling errors. They provided several suggestions using a single character rule-based error model. The error model consists of two-tape finite-state automaton mapping of any string of the error model alphabet to at least one string of the language model alphabet. Llobet et al. [LNCPCA10] proposed a similar error model using OCR to improve the recognition results of handwritten Spanish in scanned forms.

As described in Section 3.1, the single character rules are extracted using Levenshtein edit distance algorithm [HNH08]. The primitive operations: insertion, deletion, and substitution are used to represent the rules. These rules are used for constructing the transducers and editing the strings. The single rule is held on one transition in the transducer. To insert the character $f$ in a string, the insertion rule is used as $\varepsilon \rightarrow f$. The deletion of character, for example $r$, is presented by $r \rightarrow \varepsilon$. The substitution operation is done by the rule, such as $f \rightarrow t$.

## 6.2 Context-Dependent Error Model Approach

In this section, extracting the context-dependent rules and constructing the error model transducer using those rules are described. The language model and the alignment technique are explained.

### 6.2.1 Context-Dependent Confusion Rules Extraction

The error model is used to refine the candidate corrections and process the strings which do not exist in the dictionary. The goal is to sift and correct the mistakes of the recognition outputs.

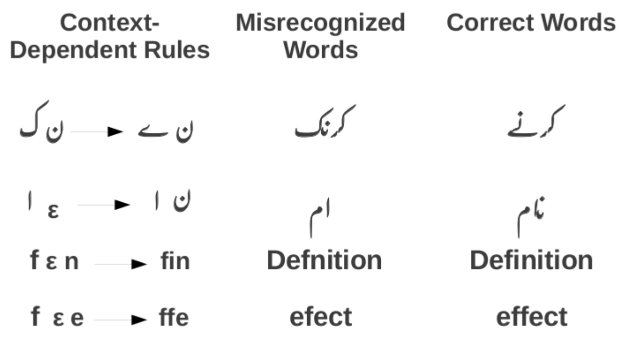


Figure 6.1: Context-dependent rules which are required to fix the misrecognized words and their corresponding correct words. $\varepsilon$ corresponds to insert a character in this position.

The error model returns a small selection of the best matches for the language model. When the suggested corrections are defined, the corrections are ranked based on the given likelihood. The error model is built using the Levenshtein edit distance algorithm [Lev66a].

The misrecognition is adapted to be used as a number of operations applied to the characters of a string: deletion, insertion and substitution with the neighbor characters on the leftmost and rightmost sides as context. The context of charaters is combined as a rule. The characters are dependent and involved together in the composition of the rule with the strings. The size of the context can be controlled, as shown in Figure. 6.1. For example, the misrecognized word $Defnition$ which needs the rules $f\varepsilon n \rightarrow fin$ to be fixed. The misrecognized word $efect$ requires the rule $f\varepsilon e \rightarrow ffe$, as shown in Figure 6.1. The error model is a transducer and is constructed by aligning the misrecognized word of the OCR output with their corresponding ground truth. By using the outputs of the alignment, the OCR confusions are extracted in the form of rules to be used in the error model with respect to their context in both misrecognized, and ground truth wordforms. The error model is used as described in Section 3.2.1 in Chapter 3. In this Chapter, new rules are extracted for Urdu script, as shown Figure 6.1

### 6.2.2 Constructing the Error Model using Weighted Finite-State Transducers (WFSTs) and the Alignment Technique

The extracted context-dependent rules in Section 6.2.1 are used to construct the error model transducer. The context-dependent rules have two parts, the left part is the OCR confusions and the right part is the corresponding corrections which are driven from the ground truth. The context-dependent rules have probabilities. The probabilities are derived from the confusion matrix of the OCR classifier and considered as costs of the rule to be used in the ranking. To use the rules with their cost, the error model transducer is built as a WFST. The weight represents the cost of the rule and input/output labels represent the left/right sides of the rules

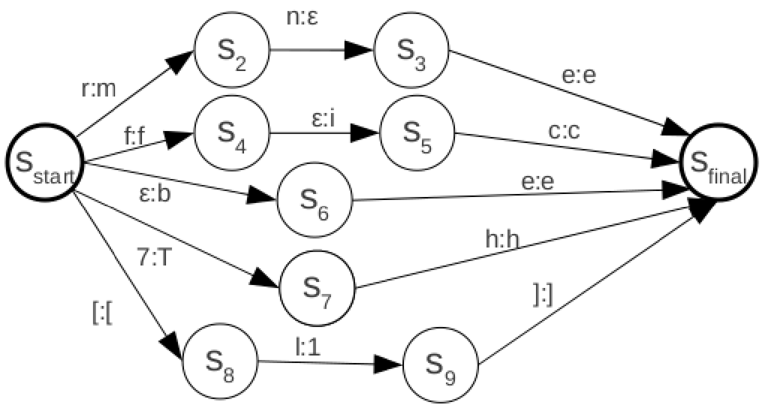


Figure 6.2: Sample of the Latin context-dependent confusion rules in error model transducer

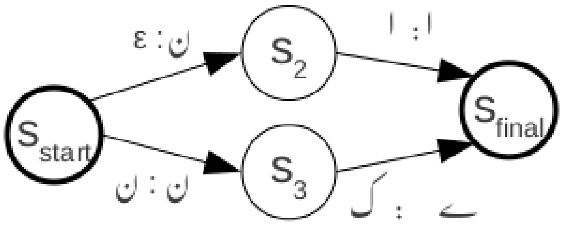


Figure 6.3: Sample of the Urdu context-dependent confusion rules in error model transducer

which help to improve the misrecognized string into the correct string.

Therefore, the error model transducer is able to map the OCR error by matching the output label of the OCR transducer with the input label of the error model. The composition is done by taking the output labels of the error model and matching them with the corresponding input label of the dictionary. Later, the OCR errors are adjusted to their correspondence corrections. Part of the constructed error model transducer for Urdu is shown in Figure 6.3 and Figure 6.2 for English. The simple operations to add a transition or state are used from the open-source OpenFST library. The newly designed error model using WFSTs for this thesis is described in [AAB14a,AAUHLB14] and has achieved a competitive performance for building and applying the context-dependent rules using WFSTs.

Three transducers are aligned. First, the OCR outputs are aligned with the error model to generate a composed Levenshtein transducer with OCR confusions of OCR output. The alignment technique is shown in Al Azawi et al. [AALB13] and is described in Section 3.2.3. The composed transducer has all the appropriate candidate corrections of the misrecognized tokens and supports the language model by providing the suggestions to decide which token is the best candidate correction. After the first alignment, the produced WFST is aligned with the dictionary to filter out words that do not exist in the language. The aligned WFST has many paths depending on the compositions of the correspondence rules in the error model. The best path with lowest cost is chosen from the second composition.

## 6.3 Character-Level Correction and LSTM Neural Networks Approach

This Section shows the powerful, novel contribution of this thesis for the task of language modeling corrections using learning-based techniques which is newly contributed in the field with simple novel designed pre-processing alignment technique. The RNN with LSTM is used to solve the problem of OCR corrections. The RNN has had very little attention in the past.

Levenshtein edit distance technique is applied to align the training pairs. The alignment technique of two strings finds the similarities and differences between them and can be interpreted as point mutations. If they share common characters, mismatches or gaps then insertion or deletion mutations are introduced in one or both lineages.

### 6.3.1 Character-Epsilon Alignment

In this section, a preprocessing method is described to allow insertion, deletion, and substitution operations in the wordform. An epsilon is inserted after each character of the misrecognized wordform. Then, the difference between two sequences is measured to find the minimum number of edits operations. If they match, then there is no change. If insertion, then the character in the second sequence is aligned with a gap. If deletion, then the character in the first sequence is aligned with a gap. In substitution, the character is replaced.

An example where substitution is required:

1. Input misrecognition: 7he
2. Adding epsilon to the input: 7 $\epsilon$ h $\epsilon$ e $\epsilon$
3. Applying alignment to obtain:
   Misrecognition = 7 $\epsilon$ h $\epsilon$ e $\epsilon$
   Ground Truth= T $\epsilon$ h $\epsilon$ e $\epsilon$

Another example where insertion is required:

1. Input misrecognition: voic
2. Adding epsilon to the input: v $\epsilon$ o $\epsilon$ i $\epsilon$ c $\epsilon$

Table 6.1: Error rate of the implemented Character-Epsilon LSTM model, Context-Dependent Confusion Rules WFTS compared to the original OCR recognition results, and Single Character Rule-Based model. The LSTM is trained 100 times and reports the mean of the error rate from different training models. The Rule-Based is skipped from Urdu dataset because of low performance amongst other approaches.

| Dataset | OCR | Single Character Rule-Based | Context-Dependent Confusion Rules WFTS | Character-Epsilon LSTM |
|---|---|---|---|---|
| English Testset | 1.14% | 1.0% | 0.68% | 0.48% |
| Urdu Testset | 6.9% | - | 3.8% | 1.58% |

3. Applying alignment to obtain:
   Misrecognition = v $\epsilon$ o $\epsilon$ i $\epsilon$ c $\epsilon$
   Ground Truth= v $\epsilon$ o $\epsilon$ i $\epsilon$ c e

### 6.3.2 String Encoding

A sequence of vectors represents the string. The encoding is done by usage of the code point for presenting each character in the string. Misrecognized wordforms along with their transcriptions are fed to the network, which performs the forward propagation step first. Alignment of output with associated transcriptions is done in the next step, then finally the backward propagation step is performed.

The LSTM is like a computer memory cell providing three multiplicative gates, namely input, output, and forget gate in order to simulate write, read, and reset operations. LSTM can be used to remember contexts over a long period of time. In order to be aware of the context in both directions, a variant name BLSTM is introduced by [GLF$^+$09]. Furthermore, a CTC layer has been introduced to overcome the limitation of data pre-segmentation.

Pre-processing and the feature vectors, e.g., characters and epsilons, are newly presented in this thesis for the task of language modeling corrections and alignments. Further explanations and network configurations are available in the experiments (Section 6.4).

After each epoch, training and validation errors are computed and the best results are saved. When there is no significant change in validation errors for a pre-set number of epochs, the training is stopped. There are two parameters which need to be tuned; namely number of hidden states and the learning rate.

## 6.4 Experiments

The approaches are evaluated on two different scripts, to show that the designed methods are language independent. One is Latin script (English) and the second is Urdu Nastaleeq (Arabic). All approaches are implemented in C++ and Python under Linux.

| OCR | Rule-Based | LSTM | Ground Truth |
|---|---|---|---|
| کروی | کردی | کردی | کردی |
| علی | علی | علی | علی |
| کربھی | کرہی | کرہی | کرہی |
| مسلیک | مسلک | مسلک | مسلک |

Figure 6.4: Sample result of LSTM and Rule-based models with the correspondence OCR and Ground Truth.

### 6.4.1 OCR Datasets UWIII and Urdu

Two datasets of recognized script have been used for the experiments, the first is based on English script, and the second is based on Urdu. The English script dataset is extracted from the OCRed collected work which is freely available from the web[2]. The Urdu script dataset is the UPTI (Urdu Printed Text Images) dataset [UHBAR+13], which contains synthetic scanned image data. Various degradation techniques are applied to increase the size of dataset. In the recognition phase, two parameters namely the number of hidden states 100, and learning rate 0.0001 are evaluated for their respective effect on the recognition accuracies. Parameter selection is done for a case where the ligature shape variations (191 classes) are considered.

In the experiments, the datasets are divided into training and testing sets as described in [UHBAR+13]. The OCR's output is used with its corresponding ground truth to build the LSTM and the error model. For the Urdu datasets (UPTI), 60,177 entries are used for training purposes, and 8,376 entries for testing purposes.

In the experiments using English script, 6,000 training inputs with their corresponding ground truth are used. In testing, 3,917 testing inputs are tested, and their corresponding UTF-8 encoded text format are used as ground truth to evaluate the generated by the proposed approaches.

### 6.4.2 Experimental Results

Preparation of the alignment technique is described in Section 6.3.1, which is considered a pre-processing step. The context-dependent EM has 830 extracted context-dependent rules to build the error model. The rules are used to build the error model using WFST. The language

[2] OCRopus Project: `https://github.com/tmbdev/ocropy`

model can be as simple as a list of finite vocabularies compiled into probabilistic finite-state transducers. The vocabularies are extracted with their frequencies from UPTI and English text corpus. The standard WFSTs framework include probability estimates for constructing a unigram model. The error rate is measured using edit distance to find the number of the edit operations on the character level. Figure 6.4 shows samples of the results of LSTM and rule-based models on both datasets.

Table 6.1 shows the evaluation of the approaches using the whole testset. The unknown wordforms are a subset from the testset. The evaluation shows an effective performance of the implemented LSTM. The context-dependent EM approach of English script has 162 rules. The single character rule-based approach of English script has 237 rules. An example is shown in the correction of the misrecognized words “rnethod” to “method” and “artifcial” to “artificial”.

The configuration of the network and the number of weights mapping between and within layers is shown in [GLF+09]. Training of the network proceeds by choosing text input lines randomly from the training set, performing a forward propagation step through the LSTM and output networks, then performing forward-backward alignment of the output with the ground-truth, and finally performing backward propagation.

LSTM networks are trained with hidden-states of different sizes 40, 60, 80, 100, 120, 140, and 160. The optimal error rates are obtained when the size of the hidden-states is in a range between 100 and 160. Time increases linearly with an increase in the number of the hidden-states. After the model has been trained, the predictions are fast. The most appropriate number of hidden-states is determined at a learning rate of 0.0001.

## 6.5 Discussion

In this chapter, two new methods of building a language model to correct OCR errors have been discussed, one based on WFST, the other based on LSTM. The experimental results show that the implemented methods achieve improvements of the OCR results when being compared to single character rule-based models. The LSTM based method yields the best performance. There are several indications that the LSTM-based approach generalizes much better to unseen samples than WFST and other approaches proposed in the literature.

The Wordlist approach, for example, is unable to process unseen samples. The rule-based approach might not be able to cover all the misrecognized variants when the OCR does recognition errors which did not appear during training. In summary, most existing approaches just pass unseen samples, leaving them unchanged. Both implemented methods process all tokens in the testset.

The LSTM model is able to predict all different misrecognized variants accurately and significantly better than WFST. The implemented approaches have no limitation on the word length and the number of errors that occur in the words. The approach is completely language independent and can be used with any language that has a dictionary and text data to build a language model.

# Chapter 7

# Combination of Multiple Recognition Systems

The purpose of this Chapter is to improve the results by designing new strategy of integrating multiple recognition outputs of recognizers. Such an integration can give higher performance and more accurate outputs than a single recognition system. The problem of aligning various OCR results lies in the difficulties to find the correspondence on character, word, line, and page level. These difficulties arise from segmentation and recognition errors which are produced by OCRs. Therefore, alignment techniques are required for synchronizing the outputs in order to compare them. For these alignment techniques, many OCR outputs were combined by aligning text lines from the first OCR output with text lines from the other OCR output.

For example, Multiple Sequence Alignment [WYM13], and heuristic search [RJN96] approaches correct recognition errors which have different intensity and type in the aligned text. These approaches fail when the same error occurs in all sequences of the OCRs. If the corrections do not appear in one of the OCRs, then none of the approaches are unable to improve upon the results of the OCRs. A Line-to-Page alignment with edit rules using WFSTs is designed. These edit rules are based on the edit operations: insertion, deletion, and substitution.

Therefore, an approach is designed using RNN with LSTM to predict these types of errors, and to solve the mentioned problems as shown in Figure 7.1. A new novel method is designed to normalize the size of the strings for the LSTM alignment. The LSTM returns best voting, especially when the heuristic approaches are unable to vote amongst various OCR engines. LSTM predicts the correct characters, even if the OCR could not produce these characters in the outputs. The contributions of this Chapter are:

C1 Line-to-Page Alignment approach as described in Section 7.2

(a) Solving the problem of segmentation and various line breaks by aligning line to page.

(b) Solving the problem of the recognition error by using edit operations as rules. The usual edit operations such as insertion, deletion, and substitution are used.

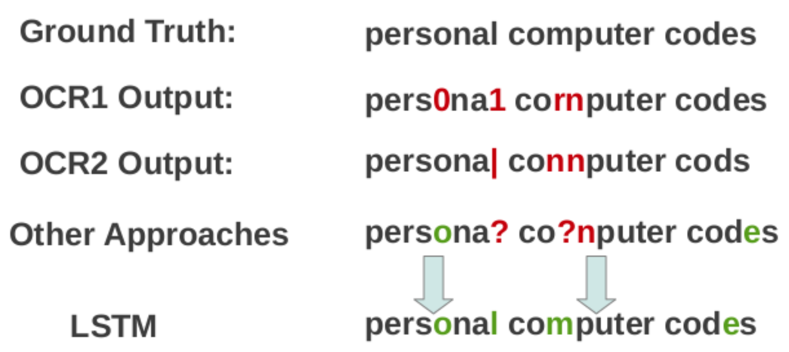


Figure 7.1: Results of the OCR Systems, LSTM alignment, and other state-of-the-art approaches such as heuristic search and Pairwise of Multiple Sequence alignment.

(c) Flexible combination and adaptation using WFST.

(d) Improving recognition results using multiple outputs from different recognition systems without using dictionary.

C2 LSTM approach as described in Section 7.3

(a) Normalization of the strings of different OCR outputs.

(b) Yields prediction of unknown strings and can vote for the best output amongst various errors.

(c) Solves all problems, which are not solved by previous approaches. Heuristic approaches are unable to return characters which did not appear in the OCR results. They also fail to vote for a correct character, if all the OCR systems provide the misrecognized versions of this character.

(d) Flexible and adaptable approach.

(e) Improved the recognition results by combining the output of many OCRs.

(f) Approaches in C1 and C2 are language independent.

(g) Performance is better than the state-of-the-art [RJN96, WYM13].

(h) The approaches are evaluated on OCRs output from the English UWIII and historical German Fraktur dataset which are obtained from state-of-the-art OCR systems.

(i) Experiments show that the error rate of ISRI-Voting is 1.45%, the Pairwise of Multiple Sequence is 1.32%, the Line-to-Page alignment is 1.26%, and the LSTM approach has the best performance with 0.40% for English script from the UWIII dataset.

The rest of the chapter is structured as follows. In Section 7.1, the state-of-the-art Pairwise of Multiple Sequence Alignment and ISRI OCR voting tools are described. Section 7.2 shows the contributed method and the constructing of Line-to-Page Alignment using WFSTs. Section 7.3 explains the second novel contributed approach using alignment and LSTMs. In Section 7.3.1, the newly designed Character-Epsilon alignment for size normalization is shown. Section 7.3.2 describes the string encoding and Section 7.3.3 shows the configuration of the LSTM network. Section 7.4 presents the experimental results. Section 7.5 summarizes conclusion.

## 7.1 State of the Art Methods

In this Section, two state-of-the-art methods are described. The two methods are used in the experiments of this Chapter and compared to newly implemented methods.

### 7.1.1 Pairwise of Multiple Sequence Alignment

Wemhoener et al. [WYM13] evaluated an alignment technique for combing OCRs outputs from scanned books. They generated the pairwise alignments of the recognized text output of the books. A pivot is chosen as the best OCR output. The pivot is aligned with each single OCR output separately to find the corresponding sequences on character level and also to fill the gaps by using *null* character. Then, output sequences in these pairwise alignments are aligned with each other to find the best output. This method can reduce the OCR error as long as the same error does not occur over multiple scans.

In the work of Wemhoener et al. [WYM13], the *null* symbol which is represented by @ is used when a character in one text fails to align with a character in the other text. In some books, the input text are differentiated in various editions, and big text portions might be missing from one edition to another. Using *null* symbol adds irrlevant characters in the alignment and makes the alignment slower.

Unaligned Texts:

String 1 = had I expressed the agony I frequentl felt he would have been taught

String 2 = had I sed the agony I fefjuently felt he would have

Pair-Wise Alignment:

String 1 = had I expressed the agony I fre@@quentl@ felt he would have been taught

String 2 = had I @@@@@s@ed the agony I f@efj@uently felt he would have@@@@@@@@@@@@

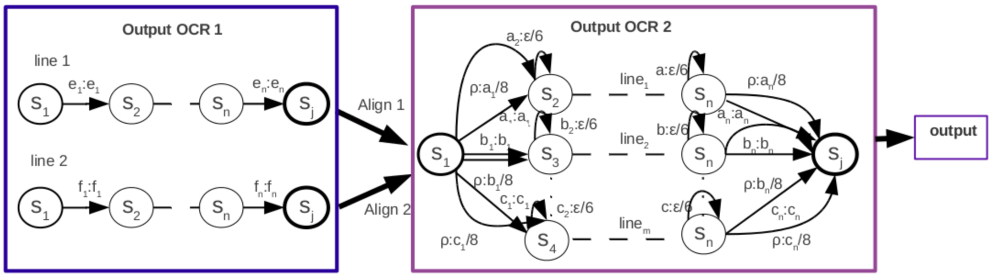


Figure 7.2: Illustrate Line-to-Page Alignment Approach. The Figure shows the alignment of each line of the first OCR with the page of the second OCR system. It shows the representation of the line and page in WFSTs, [a-f] are the input/output labels, $\rho$ label is used for substitution, and the label $\varepsilon$ is used for insertion or deletion.

### 7.1.2 ISRI OCR Voting Tool

The Information Science Research Institute (ISRI) at the University of Nevada, Las Vegas (UNLV) conducted evaluations of OCR systems [RJN96, ISR10]. The OCR systems that were evaluated are known as page readers. ISRI has been actively developing performance measures for page-reading systems. The system remains in use [ISR10].

The measures include character accuracy, marked character efficiency, word, non-stopword, and phrase accuracy as well as the the cost of correcting automatic zoning errors. Their vote program applies a voting algorithm to produce a more accurate single text file. The inputs for the voting program are the outputs of other OCR systems. The input files are first aligned so that agreements and disagreements amongst the page readers are evident. A majority vote is then taken to resolve the disagreements. The algorithm involves heuristics. The resulting text is then written. The accuracy of the voting output is normally greater than the accuracy of the text from each page reader. The tool has an option to enable some important optimizations and should be specified to get the best results. An output character is marked as a suspect if it receives no more than the fraction of votes specified by an option. Another parameter indicates the fraction of a vote that each input character receives if it is marked as being a suspect. This reduces the influence of marked characters on the voting.

Voting:

String 1 = had I expressed the agony I frequentl felt he would have been taught

String 2 = had I            sed the agony I fefjuently felt he would have

## 7.2 Line-to-Page Alignment using Weighted Finite-State Transducers Approach

A Line-Page approach is designed to avoid the difference in the lines' order for both OCR systems, where each line in OCR's page is represented as a line in the WFST. The OCR's page is represented as parallel lines for the page WFST. This approach aligns the line of the first OCR with each line in the page WFST of the second OCR. Best matches between the first OCR and the second OCR systems are represented in a composed graph. From the composed graph, the best valid path is then chosen.

Figure 7.2 shows the idea of the Line-to-Page alignment approach which separately aligns each line in the first OCR system with the whole page output of the second OCR system.

The page WFST consists of $j$ states and $m$ lines. Each line is represented as a string and each character is considered as input and output label for the transition between all pairs of two states. Considering $a_1$ as the input as well as the output label for the first character in the line, this representation is followed until the last character is reached in the line which is $a_n$, as shown in Figure 7.2.

The same representation is followed for all subsequent lines. All the WFST lines of one page begin in the same start state and end in the same final state. The OpenFST library [CAM07] provides further information regarding the basic definition of WFST.

Al Azawi et al. [AALB13] designed an alignment approach using WFSTs to align the OCR lattices with different edition transcription. To solve the problem of recognition and segmentation errors, the input of the Line-to-Page WFSTs is adapted to be used for aligning text line strings from the first OCR with the whole text line strings from the corresponding page of the second OCR. The designed approach helps to solve the over and under segmentation problems in OCR which disturb the order of the lines in a page.

In the approach, the edit operations are substitution, insertion, and deletion. They are used to smooth the alignment between the output of the OCR systems. By using them, the WFSTs provide flexibility and adaptability to process a large array of OCR processing options. They are translated as transitions in the WFST and used as rules to substitute, insert, and delete characters from the OCR. Those editing rules are: Substitution: $\rho \rightarrow a$, Deletion: $\rho \rightarrow \varepsilon$, and Insertion: $\varepsilon \rightarrow a$, where $a$ is a character, $\rho$ is *Rho* and $\varepsilon$ is *epsilon*. The cost of those edit operations is empirically estimated. The matchers are used to find and iterate through requested labels at WFST states; each matcher has a principal that is used in composition matching, as described in Chpater 4. They may implement matching special symbols that represent sets of labels such as $\rho$ (rest) which can be used for more compact automata representations and faster matching. If there is no symbol in the OCR text line that match the transcription characters, then the Rho matcher is used.

## 7.3 Long Short-Term Memory Alignment Approach (LSTM)

A new technique is designed based on LSTM recurrent networks to solve the problem of voting for the best candidate correction amongst various OCR systems. It predicts the missing characters from both OCR systems. It votes for the correct character when both OCR systems provide misrecognized characters. The initial step will be discussed in Section 7.3.1, which is the preprocessing step includes the string normalization. Following this, the features extraction is in Section 7.3.2. Finally, the LSTM network configuration is discussed in Section 7.3.3.

### 7.3.1 Preprocessing using Character-Epsilon Alignment

The preprocessing method is discussed in this section which is used to extract the features. It is used to allow insertion, deletion, and substitution operations in the strings. The strings are aligned to find the optimal alignment to give a score. The path which gives a better score represent the alignment output. The value with the possible sources (Match, Substitution, Insertion, and Deletion) is compared to see which character it came from.

To normalize the strings size, an epsilon is inserted after each character of the word of the text lines. An alignment is then applied using the Levenshtein edit distance between the words and the corresponding ground truth to obtain the optimal aligned character pairs. For example, havng the input word:

e $\epsilon$ f $\epsilon$ e $\epsilon$ c $\epsilon$ t $\epsilon$

requires an insertion for the alignment to be possible obtaining:

Input word = e $\epsilon$ f $\epsilon$ e $\epsilon$ c $\epsilon$ t $\epsilon$

ground truth word = e f f $\epsilon$ e $\epsilon$ c $\epsilon$ t $\epsilon$

Another example where substitution is required:

Input word = d $\epsilon$ i $\epsilon$ H $\epsilon$ e $\epsilon$ r $\epsilon$ e $\epsilon$ n $\epsilon$ t $\epsilon$

ground truth word = d $\epsilon$ i f f $\epsilon$ e $\epsilon$ r $\epsilon$ e $\epsilon$ n $\epsilon$ t $\epsilon$

The LSTM network is provided with the feature vector of the input and ground truth aligned pairs for training purposes as described in section 7.3.2. For testing purposes, designed LSTM trained model is applied after adding epsilon $\epsilon$ behind each character in the input word of the test token, then the correct string is predicted.

### 7.3.2 Strings Encoding

The string encoding is presented by a sequence of vectors as shown in Figure 5.3 in Section 5.2.3. The dimension is speicfied by the size of the characters set. Normalized input words along with their ground truth are fed into the network, which performs the forward propagation step first. Alignment of output with associated transcriptions is done in the next step and then finally a backward propagation step is performed.

Table 7.1: Evaluation of the experiments applied on the UW-III and German Fraktur script from Fontane dataset using OCR1, OCR2, ISRI-Voting Tool (ISRI), Line-to-Page Alignment (Line-Page WFST), and Long Short-Term Memory (LSTM) approaches. The Table shows the error rates of each approach.

| Dataset | OCR1 | OCR2 | ISRI [ISR10] | Pairwise [WYM13] | Line-Page WFST | LSTM |
|---|---|---|---|---|---|---|
| UW-III | 2.0% | 1.56% | 1.45% | 1.32% | 1.26% | 0.4% |
| Fraktur | 2.7% | 2.5% | 2.41% | 2.36% | 2.31% | 0.39% |

After each epoch, training and validation errors were computed and the best results were saved. When there is no significant change in validation errors for a pre-set number of epochs, the training is stopped. There are three parameters, which need to be tuned, namely input word size, size of the hidden-states, and learning rate. This will be explained in detail in Section 7.3.3.

### 7.3.3 LSTM Networks Configuration

LSTM has been applied to many of the language modeling tasks in this thesis and has been shown promising results. LSTM can be used to remember contexts over a long period of time. Therefore, it fits to natural language processing purposes tasks.

The LSTM networks are trained with hidden-states of different sizes 40, 60, 80, 100, 120, 140, and 160. The optimal error rates are obtained when the size of the hidden-states ranges between 100 - 160, and takes 790 - 1100 minutes for training respectively. Running times for the training is on a modern desktop PC with four cores and 8 GB RAM.

By keeping the hidden-states size fixed, the learning rate was varied between 0.001, 0.0001, and 0.00001. The most appropriate number of hidden-states was determined keeping the learning rate constant at 0.0001. After the model had been trained, the predictions were fast.

## 7.4 Experimental Results

OCR output was used with its corresponding ground truth from two sources: an open-source OCR system OCRopus[1] (OCR2) and an Multilayer Perceptrons (MLP) Classifier (OCR1). A testing set was generated from 490 OCR output pages from the UWIII datasets obtained from an evaluation set of the OCRopus project with other state-of-the-art OCR systems. The ground truth was used to evaluate the generated corrected pages by the contributed Line-to-Page Alignment, LSTM, the state-of-the-art Pairwise of Multiple Sequence Alignment [WYM13], and ISRI-Voting tool [ISR10]. Table 7.1 shows the evaluation of the approaches Line-to-Page WFST Alignment (Line-Page), LSTM compared to the original OCR recognition results and the state-of-the-art approaches applied on Fraktur and English script. The LSTM network is trained on

[1] OCRopus Project: `https://github.com/tmbdev/ocropy`

196,257 entries. All the approaches are tested on 490 pages.

The volume of “Wanderung durch die Mark Brandenburg” (1862-1889) of Heinrich Theodor Fontane is used which have been provided from the German Text Archive (DTA). The approaches are tested on OCR results of Fraktur script of 263 documents using the commercial OCR system ABBYY [2] (OCR2) and MLP (OCR1). The LSTM is trained on 75,772 entries. The LSTM Memory has the best performance of 0.40%, and 0.39%.

The error rate is measured using the Character Error Rate (CER) tool of [NRJ96]. The CER is measured using the edit distance to find the number of edit operations on character level.

$$CER = \frac{I + D + S}{N} \times 100 \tag{7.1}$$

The ISRI voting tool applies a heuristic search to align OCR systems. The tool processes each text page of the first OCR with the corresponding text page of the second OCR. The order of the OCR’s text pages in the input of the ISRI tool is important during voting procedure. The best OCR’s output should be entered first. There is no language modeling involvement. The Line-to-Page Alignment approach aligns each text line of the first OCR system with the text page of the second OCR system. The alignment is based on a character-level composition. All the characters of the OCR systems are involved in the composition. The line lattices are created from the composition in the form of WFST. The best path is then produced from the N-best paths of the composed OCRs outputs.

### 7.4.1 Analysis

In Line-to-Page WFSTs, the usage of the *epsilon* is optimized and associated with cost, while Wemhoener et al. [WYM13] has added the *null* and involved irrelevant characters in the alignment, and made the alignment slower.

The voting approaches search among the various candidate strings to find the best corrections for improving the recognition outputs. The number of the OCR systems is irrelevant, if all the systems have a high error rate, then the task of finding the best correction becomes difficult. Therefore, a new technique for learning the OCR errors is required to deliver significant improvements. It helps to come up with corrected characters for the string especially characters are not produced by the recognition systems. The LSTM model learns the recognition errors from the outputs of the OCR and is able to solve the confusion problems between the two aligned OCR outputs. For example, different substitution errors for the same character in a string. One such example is in recognizing of the character $m$ where each OCR system has misrecognized it as $rn$ and the other OCR has also misrecognized it as $nn$. An example of top confusions of the OCRs using Fraktur script are $s : f$, $c : e$, and $epsilon : i$ which are solved by LSTM, but not the other approaches. LSTM has confusions in $s : epsilon$ and $l : epsilon$.

[2]http://www.abbyy.de/

The LSTM model is able to solve the misrecognition, whilst the heuristic approach cannot. Only learning-based models predict unknown correct characters and insert them, even if both OCR systems are unable to provide them. In other approaches, the unknown characters are not produced, if not provided by both OCR systems.

## 7.5 Discussion

The Chapter presents two novel and efficient low cost methods for digitizing books with several corresponding OCR's results. The results of the OCRs combine with segmentation and recognition errors. The segmentation errors make the alignment between two recognition results not possible when the line breaks are in different positions. As a consequence, the word, line, and page correspondences are lost.

Various recognition errors in different recognition systems prevent the correspondences on character or word level. Therefore, edit rules were added to the transcription with empirical cost to smooth the alignment, and make the composition of the different transcription more compatible.

The contributed Line-Page Alignment using WFSTs provides the best alignment between OCR systems. The approach aligns each line of first OCR with the page of the second OCR to avoid the segmentation problem.

The novel contributed LSTM approach provides a significant reduction in the error rate. This is achieved by the high performance of the neural networks prediction to find the appropriate voting for the right candidate corrections.

The LSTM learns from the OCRs errors and is able to choose the best correction from the multiple aligned candidates. The other approaches fails, if the correct character does not appear in one or all of the OCRed editions. Also LSTM returns the appropriate correction better than the others, if the majority choice of the correction is not found in the OCRed editions, and the other approaches can not return the right decision. LSTM also predicts the correct votes for unknown characters and tokens.

The method offers alignment of a flexible number of recognition systems. It is generally language-independent. Therefore, there is a plan to align more recognition systems. The designed method will also be applied on various scripts such as Urdu. Because it is challenging and difficult to obtain high accuracy results from one recognition system, this approach could potentially be applied to historical documents and handwriting recognition.

# Chapter 8

# Discussion and Future Work

This thesis has achieved several key contributions to improve the state-of-the-art in language modeling, alignment techniques for recognition systems, and document analysis and processing.

This work provides a high-performance, high-accuracy, trainable language modeling and alignment system. The thesis aims of building general-purpose language modeling using high performance techniques for historical and modern languages. The key contributions solved many problems in the tasks of alignment of image segments with different transcription editions, alignment of multiple recognition systems, hyphenation modeling, correction of misrecognition, dehyphenation, and transcription processing. In the thesis, the implemented approaches are language-independent. They need less effort to train and can learn from few data.

Recognition systems become very important for digitizing historical and modern documents. The correct and clean recognition outputs are used for many purposes, such as, finding the corresponding transcription to the image segments on word, line, page levels, building n-grams, linguistics dictionaries, and understanding the contents of the documents.

Therefore, one of the core concern of this thesis has been the following question: Can we learn from the recognition errors to improve the results? The answer is yes, learning the recognition results assists the dictionary based approaches to find the appropriate, correct word in a very efficient manner. By observing the insertion, deletion, and substitution errors from the confusion of misrecognized tokens, these errors can be translated into edit rules.

Chapter 3 offers the method of building an error model using confusion rules with context-dependent information. The experiments showed that the error model approach performs better than the state-of-the-art single rule-based approaches [HNH08, LNCPCA10]. By using context information of the neighbour characters, the search is restricted to a certain string related to the misrecognized word. These confusion rules with context-dependent avoid the calculations, save time, and provide better accuracy than the single rule-based which involves several irrelevant rules in the composition. The designed error model also solves the problem of correcting the unknown words in the given dictionary.

Historical documents have various scripts and languages. A small amount of training samples are available. The corresponding ground truth is unattainable, thus makes the training very challenging for recognition systems. The difficult image conditions and the bad segmentation also make a correct recognition output not directly obtainable. The raised questions: Can we find the correspondence between the original image/text documents with different transcription editions on page, line, and word levels?

Therefore, new alignment strategies are required to find the image segments with their corresponding transcription, improve the results, and generate more training samples. The presented alignment approaches work with correct, incorrect ground truth, and different transcription editions which have different content in some paragraphs or the whole page. The implemented alignment approaches have an embedded hyphenation model. This adapts the lines breaks of the recognition results and the transcription due to segmentation errors. There are various versions of transcription editions, which need to be altered according to the input segments.

The experiments show that the presented alignment approaches in Chapter 4 perform better that the state-of-the-art approaches [FFFB11, YM11]. The improved recognition results, the linked image, and transcription create more training data on characters and ligatures. The alignment approaches avoid the effort of a manually generating ground truth and help to digitize and index the content of historical books with their editable format.

The research in this thesis found that the designed dehyphenation model using machine learning can be used for cleaning, removing regular, and noisy hyphens from recognition results or any noisy transcriptions. Many linguistic features are explored in this research to show the best performance, but the syntax-based models using regular expression are not good enough to solve various hyphen problems.

Using the technique from finite-state theory and avoiding the calculation, makes the approaches very fast and efficient. The WFSTs are flexible, easy to adapt, and the alignment can be done smoothly in hard situations. The WFSTs showed good performance and have been used in many applications such as speech recognition and machine translation.

The second part of this thesis is to design general-purpose language modeling using high performance Artificial Neural Network to improve, normalize, and correct the historical and modern languages. This is motivated by the demand of scholars and digital humanities to understand the contents of the old centuries historical texts and study many subjects in those documents, such as: Languages, religion, politics, and science. The purpose of Chapter 5 is to translate the historical text forms to modern text forms by normalizing the historical documents, where the language has different forms from the modern forms.

It is not possible to cope with all historical documents from the same or different centuries because of the variance in the orthographic rules of those centuries. There rules are highly contrasting and deviating from one document to another.

Therefore, the implemented novel Character-Epsilon alignment with LSTM learning-based approach in Chapter 5 returned better prediction for known and unknown tokens than the state-of-the-art rule-based approach in Bollmann et al. [BPD11a]. It avoided the limitations in the word-list dictionary approach, where such mapping of token-to-token corpus are rarely available. The LSTM learns from few training data and provides a fast generated mapping corpus for further work.

Several problems of language models have been investigated in this thesis for improving, normalizing, and cleaning the historical and modern documents which are obtained by using recognition systems.

The experiments in Chapter 6 shows the more accurate and robust performance of LSTM language model to correct the recognition results. Beside English script, LSTM returned best accuracy for the very challenging Arabic Nastaleeq script in Urdu Language.

In this thesis, the presented Context-Dependent Confusion Rules and Character-Epsilon LSTM learning-based approaches are the first and novel contribution for Urdu language modeling, and right-to-left script. Character-Epsilon learning-based using LSTM approach showed best performance in improving unseen tokens during the training. This approach is able to produce good results, even if few training samples are used.

The final question in this thesis was: How to use such methods for not only learning errors coming from one recognition system, but also learning errors using various output from multiple recognition systems? There, the same problems appear, such as, finding the correspondence on character, word, sentence, and page levels that come from segmentation errors, different text editions and misrecognition errors. The implemented language models in this thesis have been proven to robust against the inconsistent images and texts. They show a high performance with accuracy, flexibility, and are able to learn from recognition errors to improve the results. Therefore, the Line-Page Alignment using WFSTs and LSTM are used for the aligning of various transcription editions. The transcription editions have different line breaks and hyphenations under the recognition errors, diverse content, and found the correspondences of the contents between various documents. This combined model gives a powerful performance for aligning hard problematic documents and predicting errors which did not appear in the training phase. In Chapter 7, the contributed approaches of this thesis outperform the state-of-the-art approaches [RJN96, WYM13].

The correction and alignment services in this thesis offer new sources of information that open great chances for learning, aligning, and correcting of recognition systems. The services are useful for modern and historical document analysis and recognition, especially for digital humanities where such techniques are required. The services are automatic and cost saving.

This thesis is an important contribution to the historical document analysis field. It offers an simple interface for easily viewing, editing, and annotating the character and ligature segments with the corresponding aligned transcriptions. It provides a tool to view the text line images with the corresponding recognition output and ground truth. The recognition results with post-processing is provided as an OCR service.

The approaches presented in this thesis have no limitation on the word length and the number of errors that occur in the words. They are completely language independent and can be used with any language that has a dictionary and text data to build a language model.

In the future, the training purposes, the pre-processing, and feature extractions can be explored with other features, e.g., morphological properties. Various scripts and languages can be experimented with the designed approaches. The presented approaches in this thesis can be applied for other scripts, such as Chinese, Japanese, and obtained recognition results from various Arabic documents with different fonts. The presented learning-based normalizing approach for historical orthography can be used in the machine translation approach. It learns the rule of one language and translates to the other corresponding rules. Some approaches can be transferred from automatic tool to semi-automatic tools.

# Appendix A

# Preliminaries

This chapter describes the basic framework needed for Natural Language Processing (NLP) tasks, which includes operations and algorithms for constructing statistical language modeling. The chapter begins by explaining Weighted Finite-State Transducers WFSTs in Section A.1. It also provides the basic operations and a background literature review for WFSTs, such as composition in Section A.1.1, determinization, minimization in Section A.1.2, and closure in Section A.1.3. Section A.2 describes the methods and state-of-the-art of Recurrent Neural Network (RNN). Section A.2.2 is an introduction into RNN. Section A.2.3 reviews the background material for Long Short-Term Memory (LSTM), which is the main RNN architecture used in this thesis.

## A.1 Weighted Finite-State Transducers (WFSTs)

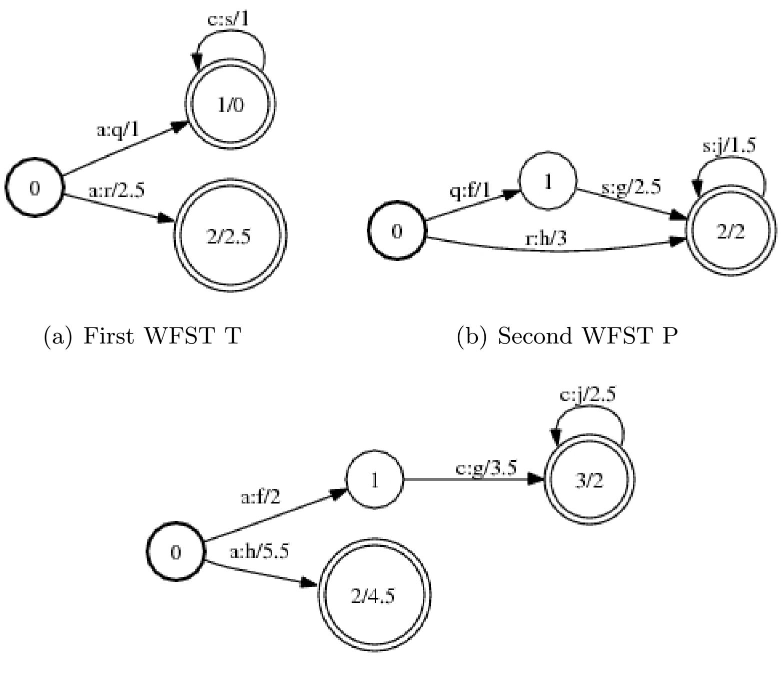
WFSTs have been used in various NLP tasks including automatic speech recognition, morphology analysis, Optical Character Recognition (OCR), spell checking, part-of-speech tagging, and machine translation.

### A.1.1 Composition

The compose operation computes the composition of two transducers. If $T$ transduces string $a$ to $q$ with weight 1 and $P$ transduces $q$ to $f$ with weight 1, then their composition transduces a string $a$ to $f$ with weight 2.

$$(0, a, q, 1, 1)\ and\ (0, q, f, 1, 2) \Longrightarrow ((0, 0), a, f, 2, (1, 2)) \tag{A.1}$$

The output labels of the first transducer or the input labels of the second transducer must be sorted. Figure A.1 shows the compose operation, as described in [CAM07]. The Figure A.1(a) shows that the first WFST $T$ starts with the start state 0 and has two final states 1 and 2. The final states can have weight, for example the final state 1 has 0 weight, and the final state 2 has



(a) First WFST T

(b) Second WFST P

(c) The composition output WFST

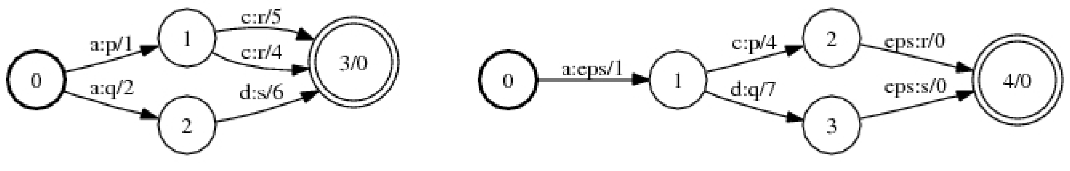
Figure A.1: Composition of two WFST (a) and (b). The result is (c)

2.5 weight. WFST $T$ has three transitions. The first transition has input label $a$, output label $q$, and weight with value 1. It starts from start state 0 to final state 1. The second transition starts from start state 0 to the final state 2. It has input label $a$, output label $r$, and weight with value 2.5. The last transition is a loop transition on the final state 1 with input label $c$, output label $s$, and weight 1.

The second WFST $P$ starts with the start state 0 and ends with one final state 2 has weight 2 as shown in Figure A.1(b). It has four transitions. The first transition has input label $q$, output label $f$, and weight with value 1. It starts from start state 0 to final state 1. The second transition starts from start state 0 to the final state 2. It has input label $r$, output label $h$, and weight with value 3. The third transition starts from state 1 to final state 2 and it has input label $s$, output label $g$, and weight 2.5. The last transition is a loop transition on the final state 2 with input label $s$, output label $j$, and weight 1.5.

The composition of the two transducers $T$ and $P$ is presented in Figure A.1(c). The composition operation is to find the matches between the output and input labels of the transitions in both transducers. For example, the transition $a : q$ in $T$ is composed with the transition $q : f$ in $P$ to output the transition $a : f$.

Second composition is between the transition $a : r$ in $T$ with the transition $r : h$ in $P$ to output $a : h$. Another composition is between $c : s$ in $T$ with the transition $s : g$ in $P$ to output $c : g$ transition. The final loop transition $c : j$ is generated from the composition of $c : s$ and



(a) Input WFST T (b) Output WFST P

Figure A.2: Determinization of WFST (a) to (b).

$s : j$. The sum of all the weights on the transitions and in the final states is used to find the best path in the WFST with low cost. The best path is a successful path from start state to the final state. The shortest-path algorithm can be used to find the $k$ lowest cost paths in a transducer Mohri et al. [Moh03c]. This allows the best string(s) to be efficiently extracted and provides a generic implementation of the maximum and minimum operations in probabilistic models.

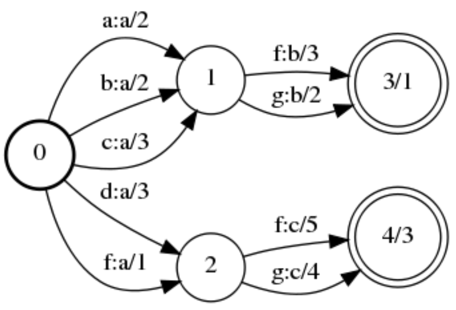
Special symbol matching [CAM07] can be used in weighted composition. Special symbols act as transition filters in composition and enable more compact topologies and faster matching. The special symbols are $\epsilon$ (epsilon), $\sigma$ (all), $\rho$ (rest), and $\varphi$ (fail). Transitions labelled $\sigma$ match and consume any arc in composition. Transitions labelled $\rho$ match and consume any arc without an explicit transition from the state. Non-consuming $\varphi$-transitions are similar to $\epsilon$ transitions but can only be taken when no regular symbol match is possible. $\varphi$-transitions are required for the exact implementation of backoff in the WFSTs representation of an n-gram language model Allauzen et al. [CAR03].

### A.1.2 Determinization and Minimization

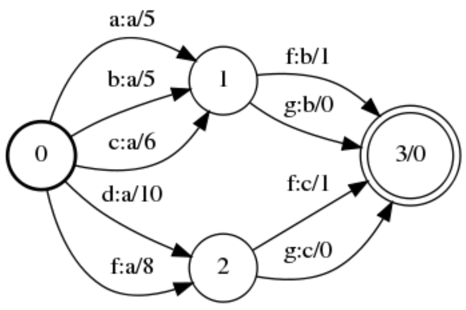
The determinization operation is used to determinize a weighted transducer. The result will be an equivalent WFST that has the property that no state has two transitions with the same input label, as shown in Figure A.2. For this algorithm, epsilon transitions are treated as regular symbols. The start state 0 in WFST $T$ has two transitions $a : p$ and $a : q$ and they have the same input label $a$. The state 1 has also two transitions $c : r$ and $c : r$ and both have the same input label $c$.

Epsilons is added as input labels at the end of the paths when determinizing transducers. If the input transducer also contains epsilons, this may result a non-deterministic results even when the epsilons are treated as regular symbols. Further explanation of the algorithm is described in [CAM07] and [CAR03].

The minimization operation performs the minimization of deterministic weighted transducers. If the input WFST $T$ is an transducer, this operation produces the minimal transducer $P$ equivalent to $T$, i.e.,the transducer with a minimal number of states that is equivalent to $T$, as shown in Figure A.3. However, this minimality is obtained by allowing transition having

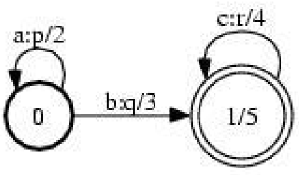


(a) Input WFST T

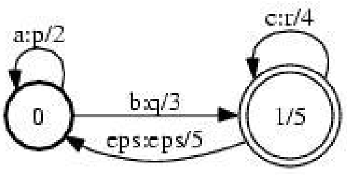


(b) Output WFST P

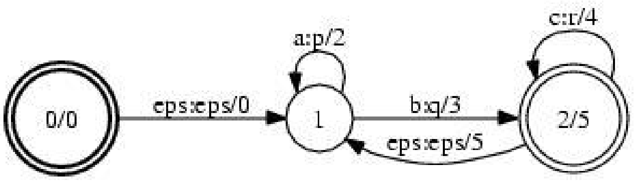
Figure A.3: Minimization of WFST (a) to (b).



(a) First WFST T



(b) Output of closure plus operation



(c) Output of closure star operation

Figure A.4: Closure plus and star operation of WFST (a) and the results are (b) and (c)

strings of symbols as output labels, this is known in the literature as a real-time transducer. The minimize function converts such a transducer by expanding each string-labeled transition into a sequence of transitions. This will result in the creation of new states, hence losing the minimality property.

### A.1.3 Closure

The closure operation computes the concatenative closure. If $T$ transduces string $a$ to $p$, then the closure transduces $a$ to $p$, $aa$ to $pp$, $aaa$ to $ppp$, etc. If the closure type is $ClosureStar$, then the empty string is transduced to itself with weight 0, as shown in Figure A.4.

## A.2 Artificial Neural Networks

Artificial Neural Networks (ANN) are similar to Biological Neural Networks in performing functions collectively and in parallel by individual units, the neurons. An artificial neuron is a computational model inspired by the natural neurons. Neural network models which emulate the central nervous system are part of theoretical neuroscience and computational neuroscience. ANNs are employed in various applications such as speech, printed characters, and handwriting recognition. McCulloch and Pitts et al. [McC] built a computational model based on mathematics and algorithms for neural networks. This model is called threshold logic.

ANN basically consist of inputs, which are multiplied by weights. Then, it is computed by a mathematical function which determines the activation of the neuron. Another function computes the output of the artificial neuron. ANNs combine artificial neurons in order to process information.

The MLP is described in Section A.2.1. The RNN is reviewed in Section A.2.2. Section A.2.3 describes the LSTM.

### A.2.1 Multilayer Perceptrons (MLP)

The MLP [Ros62] is a feedforward ANN model that maps sets of input data into a set of appropriate outputs. A MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. MLP utilizes a supervised learning technique called backpropagation for training the network. The multilayer perceptrons consists of three or more layers: an input and an output layer with one or more hidden layers of nonlinearly-activating nodes. Therefore, it considered a deep neural network. Each node in one layer connects with a certain weight to every node in the following layer. Input are entered to the input layer and the resulting unit activations are propagated through the hidden layers to the output layer. This process is called the forward pass of the network. In the hidden layers, the units have activation functions that transform the summed activation arriving at the unit.

An example MLP with $I$ input units, activated by input vector $x$. Each unit in the first hidden layer calculates a weighted sum of the input units. For hidden unit $h$, it refers to this sum as the network input to unit $h$, and denote it $a_h$. The activation function $\theta_h$ is then applied, yielding the final activation $b_h$ of the unit. Denoting the weight from unit $i$ to unit $j$ as $w_{ij}$, then:

$$a_h = \sum_{i=1}^{I} w_{ih} x_i \tag{A.2}$$

$$b_h = \theta_h(a_h) \tag{A.3}$$

The most common choices of activation function are the hyperbolic tangent:

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{A.4}$$

and the logistic sigmoid:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{A.5}$$

A further explanation of the theoretical background can be found in Graves et al. [Gra12].

### A.2.2 Recurrent Neural Networks (RNNs)

The RNN is an ANN with an additional connections from the hidden units to themselves, therefore the RNN differs from the feed forward network. The RNNs use their internal memory to process arbitrary sequences of inputs and makes the RNNs capable of solving challenging problems such as unsegmented connected handwriting recognition. The forward pass of RNN is the same as the feedforward neural network the MLP, except that the activations reach the hidden layer from both the input and hidden layer activations. As described in Graves et al. [Gra12], a length $T$ input sequence $x$ is entered to an RNN with $I$ input units, $H$ hidden units, and $K$ output units. $x_i^t$ is the value of the input $i$ at time $t$, $a_j^t$ and $b_j^t$ are the network input to units $j$ at time $t$, and the activation of the unit $j$ at time $t$ respectively for hidden unit $h$, where $h'$ is the size of the hidden unit of the recurrent connections and the weight is denoted as $w$. For the hidden units:

$$a_h^t = \sum_{i=1}^{I} w_{ih} x_i^t + \sum_{h'=1}^{H} w_{h'h} b_{h'}^{t-1} \tag{A.6}$$

Nonlinear, differentiable activation functions are then applied as for MLP:

$$b_h^t = \theta_h(a_h^t) \tag{A.7}$$

The network inputs to the output units are calculated at the same time as the hidden activations:

$$a_k^t = \sum_{h=1}^{H} w_{hk} b_h^t \tag{A.8}$$

In the backward pass, Graves et al. [Gra12] focus on Backpropagation Through Time (BPTT) [WZ95,Wer90] because it is simple and efficient in computation time. BPTT is similar to standard backpropagation. The objective function for recurrent networks depends on the activation of the hidden layer not only through its influence on the output layer, but also through its influence on the hidden layer at the next timesteps, Graves et al. [Gra12]:

$$\delta_h^t = \theta'(a_h^t)\left(\sum_{k=1}^{K}\delta_k^t w_{hk} + \sum_{h'=1}^{H}\delta_{h'}^{t+1} w_{hh'}\right), \tag{A.9}$$

where

$$\delta_j^t \overset{\text{def}}{=\!=} \frac{\partial O}{\partial a_j^t} \tag{A.10}$$

Where $\partial O$ and $\partial a_j^t$ are the derivatives of the objective function $O$ and the network input to unit $j$ at time $t$. The weights to and from each unit in the hidden layer are the same at every timestep. To obtain the derivatives with respect to each of the network weights, the whole sequnce is summed over.

$$\frac{\partial O}{\partial w_{ij}} = \sum_{t=1}^{T}\frac{\partial O}{\partial a_j^t}\frac{\partial a_j^t}{\partial w_{ij}} = \sum_{t=1}^{T}\delta_j^t b_i^t \tag{A.11}$$

For further reading regarding Bidirectional RNNs, network training, gradient descent, training with noise, input representation, and weight initialisation see Graves et al. [Gra12]

### A.2.3 Long Short-Term Memory (LSTM)

LSTM is a RNN architecture consisting of a set of recurrently connected subsets, known as memory blocks. Each block contains one or more self-connected memory cells and three multiplicative units. The multiplicative units are the input, output, and forget gates. They provide continuous analogues of write, read, and reset operations for the cells. The LSTM differs by the nonlinear units in the hidden layers of RNN are replaced by memory blocks.

In LSTM, the multiplicative gates allow the LSTM memory cells to store, and access information over long periods of time, thereby avoiding the vanishing gradient problem [HS97]. The gates determine when the input is significant enough to remember, when it should continue to remember or forget the value, and when it should output the value. Figure A.5 shows the memory block with one cell of LSTM.

LSTM has been applied to various problems, such as learning context free languages [GS01], recognition of text lines [BUHAAS13], speech recognition [GS05a, AGS06], and handwriting recognition [Gra12, GLF$^+$09]

In the work of Graves et al. [Gra12, GS05a, GS05b], the author introduces bidirectional LSTM, which is composed of the standard bidirectional RNN architecture with LSTM used in the hidden layers. BLSTM has the benefit of accessing the long range context in both input directions and outperforms other neural network architectures on sequence labelling tasks. It is widely used especially in the field of bioinformatics [TR07, CC05]. For further reading, the thesis of Graves et al. [Gra12] describes LSTM in more details.

The equations for the activation (forward pass) and gradient calculation (backward pass) of an LSTM hidden layer within a RNN are provided. The exact error gradient [GS05a], calculated

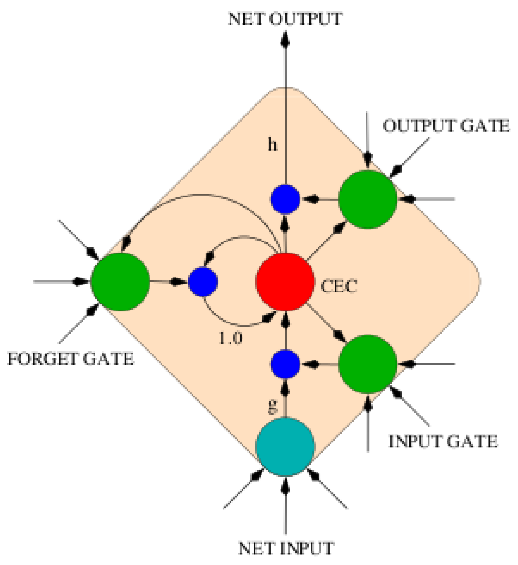


Figure A.5: LSTM memory block with one cell, as shown in Graves et al. [Gra12]

with backpropagation through time, is presented in [Gra12]. The weight of the connection from unit $i$ to unit $j$ is $w_{ij} \in \mathbb{R}$. The network input to some unit $j$ at time $t$ is represented as $a_j^t$ and $b_j^t$ is the value of the $j$ same unit after the activation function has been applied. The LSTM equations are given for a single memory block only.

For multiple blocks the calculations are simply repeated for each block, in any order. The subscripts $c$ refers to one of the $C$ memory cells. $s_c^t$ is the state of cell $c$ at time $t$. $f$ is the activation function of the gates and $g$ and $h$ are respectively the cell input and output activation functions.

The number of inputs is $I$; $K$ is the number of outputs; and $H$ is the number of cells in the hidden layer. Only the cell outputs $b_c^t$ are connected to the other blocks in the layer. The index $h$ represents the cell outputs from other blocks in the hidden layer. In LSTM, the forward pass is calculated for a length $T$ input sequence $x$ by starting at $t = 1$, as with standard RNNs, Then, the update equation is recursively applied while incrementing $t$. The BPTT backward pass is calculated by starting at $t = T$. Then, the unit derivatives are recursively calculated while decrementing $t$ [Gra12]. The final weight derivatives are found by summing over the derivatives at each timestep. $O$ is the objective function used for training. The equations calculated during the forward and backward passes should proceed as given in Graves et al. [Gra12].

# Bibliography

[AAAB13] Mayce Al Azawi, Muhammad Zeshan Afzal, and Thomas M Breuel. Normalizing historical orthography for OCR historical documents using LSTM. In *Proceedings of the 2nd International Workshop on Historical Document Imaging and Processing*, pages 80–85. ACM, 2013.

[AAB14a] Mayce I A Al Azawi and Thomas M Breuel. Context-dependent confusions rules for building error model using weighted finite-state transducers for OCR post-processing. *Proceedings 11th International Workshop on Document Analysis Systems*, pages 116–120, 2014.

[AAB14b] Mayce I A Al Azawi and Thomas M Breuel. Dehyphenation by classification for OCR results. *Proceedings 11th International Workshop on Document Analysis Systems*, 2014.

[AAKK12] Milad Alemzadeh, Kacem Abida, Richard Khoury, and Fakhri Karray. Enhancement of the rover's voting scheme using pattern matching. In *Autonomous and Intelligent Systems - Third International Conference Proceedings, Aveiro, Portugal*, volume 7326 of *Lecture Notes in Computer Science*, pages 167–174. Springer, 2012.

[AALB13] Mayce Al Azawi, Marcus Liwicki, and Thomas M. Breuel. WFST-based ground truth alignment for difficult historical documents with text modification and layout variations. In *Document Recognition and Retrieval XX (DRR)*, volume 8658, pages 865818–865818–12, 2013.

[AAUHLB14] Mayce Al Azawi, Adnan Ul Hasan, Marcus Liwicki, and Thomas M Breuel. Character-level alignment using WFST and LSTM for post-processing in multi-script recognition systems - a comparative study. In *11th International Conference on Image Analysis and Recognition, ICIAR, Vilamoura, Algarve, Portugal*, volume Part I, LNCS 8814, page 379386. Lecture Notes in Computer Science, Springer International Publishing Switzerland, 2014.

[AC09] A. Abdulkader and M.R. Casey. Low cost correction of OCR errors using learning in a multi-engine environment. In *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on*, pages 576–580, July 2009.

[AD07] Apostolos Antonacopoulos and Andy C Downton. Special issue on the analysis of historical documents. *International Journal on Document Analysis and Recognition*, 9(2):75–77, 2007.

[AGS06] F. Gomez A. Graves, S. Fern ndez and J. Schmidhuber. Connectionist a temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *In Proceedings of the International Conference on Machine Learning, ICML*, Pittsburgh, 2006.

[AM09] Cyril Allauzen and Mehryar Mohri. Linear-space computation of the edit-distance between a string and a finite automaton. *CoRR*, abs/0904.4686, 2009.

[AWH07] Klaus U. Schulz Andreas W. Hauser. Unsupervised learning of edit distance weights for retrieving historical spelling variations. In *Proceedings of the First Workshop on Finite-State Techniques and Approximate Search*, pages 1–6, 2007.

[BC11] Hildelies Balk and Aly Conteh. Impact: centre of competence in text digitisation. In *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*, HIP '11, pages 155–160, New York, NY, USA, 2011. ACM.

[BF10] Fabienne Braune and Alexander Fraser. Improved unsupervised sentence alignment for symmetrical and asymmetrical parallel corpora. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 81–89, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[BPD11a] Marcel Bollmann, Florian Petran, and Stefanie Dipper. Applying Rule-Based Normalization to Different Types of Historical Texts—An Evaluation. In *Proceedings of the 5th Language & Technology Conference (LTC 2011)*, 2011.

[BPD11b] Marcel Bollmann, Florian Petran, and Stefanie Dipper. Applying Rule-Based Normalization to Different Types of Historical Texts—An Evaluation. In *Proceedings of the 5th Language & Technology Conference (LTC 2011)*, 2011.

[BPD11c] Marcel Bollmann, Florian Petran, and Stefanie Dipper. Rule-Based Normalization of Historical Texts. In *Proceedings of the RANLP 2011 Workshop on*

*Language Technologies for Digital Humanities and Cultural Heritage*, pages 34–42, Hissar, Bulgaria, Sep. 2011.

[BR08] Alistair Baron and Paul Rayson. VARD 2: A tool for dealing with spelling variation in historical corpora. In *Proceedings of the Postgraduate Conference in Corpus Linguistics*, Birmingham, UK, 2008. Aston University.

[BRB$^+$09] Federico Boschetti, Matteo Romanello, Alison Babeu, David Bamman, and Gregory Crane. Improving OCR accuracy for classical critical editions. In *ECDL*, pages 156–167, 2009.

[Bre08] Thomas M. Breuel. The OCRopus open source OCR system. In *DRR*, page 68150, 2008.

[BUHAAS13] T.M. Breuel, A. Ul-Hasan, M.A. Al-Azawi, and F. Shafait. High-performance OCR for printed english and fraktur using LSTM networks. In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 683–687, Aug 2013.

[CAM07] J. Schalkwyk W. Skut C. Allauzen, M. Riley and M. Mohri. OpenFST: a general and efficient weighted finite-state transducer library. pages 11–23, July 2007.

[CAR03] Mehryar Mohri Cyril Allauzen and Brian Roark. Generalized algorithms for constructing statistical language models. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL'2003)*, pages 40–47, 2003.

[CC05] J. Chen and N. Chaudhari. Protein secondary structure prediction with bidirectional LSTM networks. In *Post-Conference Workshop on Computational Intelligence Approaches for the Analysis of Bio-data (CI-BIO)*, Montreal, Canada, August 2005.

[EGF06] Andrea Ernst-Gerlach and Norbert Fuhr. Generating search term variants for text collections with historic spellings. In *Proceedings of the 28th European conference on Advances in Information Retrieval*, ECIR'06, pages 49–60, Berlin, Heidelberg, 2006. Springer-Verlag.

[FFFB11] Andreas Fischer, Volkmar Frinken, Alicia Fornés, and Horst Bunke. Transcription alignment of Latin manuscripts using hidden Markov models. In *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*, HIP '11, pages 29–36, New York, NY, USA, 2011. ACM.

[FIFB11] Andreas Fischer, Emanuel Indermhle, Volkmar Frinken, and Horst Bunke. HMM-based alignment of inaccurate transcriptions for historical documents. In *ICDAR'11*, pages 53–57, 2011.

[Fis12] Andreas Fischer. Handwriting recognition in historical documents. In *Ph.D dissertation, Bern University*, 2012.

[FM06] Shaolei Feng and R. Manmatha. A hierarchical, HMM-based automatic evaluation of OCR accuracy for a digital library of books. In *Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries*, JCDL '06, pages 109–118, New York, NY, USA, 2006. ACM.

[FV11] Lenz Furrer and Martin Volk. Reducing OCR errors in Gothic script documents. In *Workshop on Language Technologies for Digital Humanities and Cultural Heritage (associated with the RANLP 2011 Conference)*, sep 2011.

[FZMEB$^+$12] V. Frinken, F. Zamora-Martinez, S. Espana-Boquera, M.J. Castro-Bleda, A. Fischer, and H. Bunke. Long-short term memory neural networks language modeling for handwriting recognition. In *21st ICPR*, pages 701–704, Nov 2012.

[GHHP97] Isabelle Guyon, Robert M. Haralick, Jonathan J. Hull, and Ihsin Tsaiyun Phillips. Data sets for OCR and document image understanding research. In *In Proceedings of the SPIE - Document Recognition IV*, pages 779–799. World Scientific, 1997.

[GLF$^+$09] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for improved unconstrained handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(5):855–868, 2009.

[GR98] Andrew R. Golding and Dan Roth. A winnow-based approach to context-sensitive spelling correction. *CoRR*, cs.LG/9811003, 1998.

[Gra12] Alex Graves. *Supervised Sequence Labelling with Recurrent Neural Networks.* Textbook, Studies in Computational Intelligence, Springer, 2012.

[GS01] F. A. Gers and J. Schmidhuber. LSTM recurrent networks learn simple context free and context sensitive languages. In *IEEE Transactions on Neural Networks*, volume 12, page 13331340, 2001.

[GS05a] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. In *Neural Networks*, volume 18, pages 602–610, June/July 2005.

[GS05b] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. In *Proceedings of the 2005 International Joint Conference on Neural Networks*, Montreal, Canada, 2005.

[GT94] Gregory Grefenstette and Pasi Tapanainen. What is a word, what is a sentence? problems of tokenization. pages 79–87, 1994.

[Gus97] Dan Gusfield. Algorithms on strings, trees and sequences. 1997.

[HB05] Graeme Hirst and Alexander Budanitsky. Correcting real-word spelling errors by restoring lexical cohesion. *Nat. Lang. Eng.*, 11(1):87–111, March 2005.

[HM11] Iris Hendrickx and Rita Marquilhas. From Old Texts to Modern Spellings: An Experiment in Automatic Normalisation. *JLCL*, 26(2):65–76, 2011.

[HNH08] Ahmed Hassan, Sara Noeman, and Hany Hassan. Language independent text correction using finite-state automata. International Joint Conference on Natural Language Processing, 2008.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997.

[ISR10] ISRI. ISRI-OCR-evaluation-tools code and data to evaluate OCR accuracy originally from UNLV-ISRI, 2010. Accessed: 2010.

[Jur10] Bryan Jurish. More than words: Using token context to improve canonicalization of historical German. *Journal for Language Technology and Computational Linguistics*, 25(1):23–39, 2010.

[KB03] Shankar Kumar and William Byrne. A weighted finite-state transducer implementation of the alignment template model for statistical machine translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 63–70, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[KMA04] E. Micah Kornfield, R. Manmatha, and James Allan. Text alignment with handwritten documents. In *Proceedings of the First International Workshop on Document Image Analysis for Libraries (DIAL'04)*, DIAL '04, pages 195–, Washington, DC, USA, 2004. IEEE Computer Society.

[KN95] R. Kneser and H. Ney. Improved backing-off for m-gram language modeling. In *Proceedings of the 1995 International Conference on Acoustics, Speech, and Signal Processing, 1995. ICASSP-95*, volume 1, pages 181–184, 1995.

[Kuk92] Karen Kukich. Techniques for automatically correcting words in text. *ACM Comput. Surv.*, 24(4):377–439, December 1992.

[LB02] Edward Loper and Steven Bird. NLTK: The natural language toolkit. In *In Proc. of the ACL on Effective Tools and Methodologies for NLP*, 2002.

[LBK$^+$99] Zhidong Lu, Issam Bazzi, Andras Kornai, John Makhoul, Premkumar Natarajan, and Richard Schwartz. A robust, language-independent OCR system. volume 3584, pages 96–104, 1999.

[LdFPeSdAF11] Rafael Dueire Lins, Gabriel de F. Pereira e Silva, and Andrei de A. Formiga. Histdoc v. 2.0: enhancing a platform to process historical documents. In *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*, HIP '11, pages 169–176, New York, NY, USA, 2011. ACM.

[Lev66a] V. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Soviet Physics-Doklady*, 10(8):707–710, 1966.

[Lev66b] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. Technical Report 8, 1966.

[LNCPCA10] Rafael Llobet, J.Ramon Navarro-Cerdan, Juan-Carlos Perez-Cortes, and Joaquim Arlandis. Efficient OCR post-processing combining language, hypothesis and error models. In EdwinR. Hancock, RichardC. Wilson, Terry Windeatt, Ilkay Ulusoy, and Francisco Escolano, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, volume 6218 of *Lecture Notes in Computer Science*, pages 728–737. Springer Berlin Heidelberg, 2010.

[Lop09] Daniel Lopresti. Optical character recognition errors and their effects on natural language processing. *International Journal on Document Analysis and Recognition (IJDAR)*, 12(3):141–151, 2009.

[LR11] W.B. Lund and E.K. Ringger. Error correction with in-domain training across multiple OCR system outputs. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 658–662, Sept 2011.

[LS12] Dar-Shyang Lee and Ray Smith. Improving book OCR by adaptive language and image models. In *Proceedings of 2012 10th IAPR International Workshop on Document Analysis Systems*, pages 115–119, 2012.

[LWR11] W.B. Lund, D.D. Walker, and E.K. Ringger. Progressive alignment and discriminative error correction for multiple OCR engines. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 764–768, 2011.

[LZ97] Daniel Lopresti and Jiangying Zhou. Using consensus sequence voting to correct OCR errors. *Computer Vision and Image Understanding*, 67(1):39 – 47, 1997.

[McC] Warren; Walter Pitts McCulloch. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 4(5):115133.

[MCL07] Christophe Hancart Maxime Crochemore and Thierry Lecroq. Algorithms on strings. 2007.

[MEAMA11] Ines Ben Messaoud, Haikal El Abed, Volker Märgner, and Hamid Amiri. A design of a preprocessing framework for large database of historical documents. In *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*, HIP '11, pages 177–183, New York, NY, USA, 2011. ACM.

[MKB$^+$11] Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic*, pages 5528–5531, 2011.

[Moh03a] Mehryar Mohri. Edit-distance of weighted automata. In Jean-Marc Champarnaud and Denis Maurel, editors, *Implementation and Application of Automata*, volume 2608 of *Lecture Notes in Computer Science*, pages 1–23. Springer Berlin Heidelberg, 2003.

[Moh03b] Mehryar Mohri. Edit-distance of weighted automata. In Jean-Marc Champarnaud and Denis Maurel, editors, *Implementation and Application of Automata*, volume 2608 of *Lecture Notes in Computer Science*, pages 1–23. Springer Berlin Heidelberg, 2003.

[Moh03c] Mehryar Mohri. Edit-distance of weighted automata. In *Seventh International Conference on Automata (CIAA 2002),Tours, France*, volume 2608, pages 1–23. Lecture Notes in Computer Science, Springer, Berlin-NY, 2003.

[NRJ96] T. A. Nartker, S. V. Rice, and F. R. Jenkins. OCR accuracy: Unlvs fifth annual test. Technical Report 96-01, Inforamtion Science Research Institute, University of Nevada, Las Vegas, 1996.

[NSD$^+$11] Clemens Neudecker, Sven Schlarb, Zeki Mustafa Dogan, Paolo Missier, Shoaib Sufi, Alan Williams, and Katy Wolstencroft. An experimental workflow development platform for historical document digitisation and analysis. In *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*, HIP '11, pages 161–168, New York, NY, USA, 2011. ACM.

[Ofl96] Kemal Oflazer. Error-tolerant finite-state recognition with applications to morphological analysis and spelling correction. *Comput. Linguist.*, 22(1):73–89, March 1996.

[ON03] Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Comput. Linguist.*, 29(1):19–51, mar 2003.

[PH12] Tommi A. Pirinen and Sam Hardwick. Effect of language and error models on efficiency of finite-state spell-checking and correction. In *Proceedings of the 10th International Workshop on Finite-State Methods and Natural Language Processing*, pages 1–9, Donostia–San Sebastian, July 2012. Association for Computational Linguistics.

[Pil05] Andrea Ernst-Gerlach; Thomas Pilz. Rsnsr: Rule based search in text databases with nonstandard orthography. Talk at Lancaster University, Lancaster, U.K., November 2005.

[Pis02] Jakub Piskorski. Finite-state machine toolkit. Technical report, DFKI, 2002.

[PSL12] Tommi Pirinen, Miikka Silfverberg, and Krister Linden. Improving Finite-State SpellChecker Suggestions with Part of Speech N-Grams. 2012.

[Rab89] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *IEEE*, volume 77, page 257286, 1989.

[RHSM07] Christoph Ringlstetter, Max Hadersbeck, Klaus U. Schulz, and Stoyan Mihov. Text correction using domain dependent bigram models from web crawls. In Proceeding of the International Joint Conference on Artificial Intelligence IJCAI Workshop on Analytic for Noisy Unstructured Text Data, 2007.

[RJN96] S.V. Rice, F.R. Jenkins, and T.A. Nartker. The fifth annual test of OCR accuracy. Technical Report 96-01, Inforamtion Science Research Institute, University of Nevada, Las Vegas, April 1996.

[Ros62] F. Rosenblatt. *Principles of neurodynamics: perceptrons and the theory of brain mechanisms.* Report (Cornell Aeronautical Laboratory). Spartan Books, 1962.

[SM02] Klaus Schulz and Stoyan Mihov. Fast string correction with levenshtein-automata. *INTERNATIONAL JOURNAL OF DOCUMENT ANALYSIS AND RECOGNITION*, 5:67–85, 2002.

[Smi07] Ray Smith. An overview of the tesseract OCR engine. In *Proc. Ninth Int. Conference on Document Analysis and Recognition (ICDAR)*, pages 629–633, 2007.

[Smi11] Ray Smith. Limits on the application of frequency-based language models to OCR. In *Proceedings of the 2011 International Conference on Document Analysis and Recognition*, ICDAR '11, pages 538–542, Washington, DC, USA, 2011. IEEE Computer Society.

[T B94] T Breuel. Language Modeling for a Real-world Handwriting Recognition Task. In *AISB Workshop on Computational Linguistics for Speech and Handwriting Recognition, 1994*, 1994.

[T B95] T Breuel. Recognition of Handwritten Responses on us Census Forms. In *Proceedings of the International Association for Pattern Recognition Workshop (Document Analysis Systems*, pages 237–264, 1995.

[TMC11] Stefan Kombrink Lukas Burget Tomas Mikolov, Anoop Deoras and Jan Cernocky. Empirical evaluation and combination of advanced language modeling techniques. In *Proc. of International Speech Communication Association*, Florence, Italy, 2011.

[TR07] T. Thireou and M. Reczko. Bidirectional long short-term memory networks for predicting the subcellular localization of eukaryotic proteins. In *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, volume 4, page 441446, 2007.

[UHBAR$^+$13] A. Ul-Hasan, S. Bin Ahmed, F. Rashid, F. Shafait, and T.M. Breuel. Offline printed urdu nastaleeq script recognition with bidirectional LSTM networks. In *12th Intern. Conf. on Document Analysis and Recognition*, pages 1061–1065, 2013.

[Vit67] Andrew Viterbi. Error bounds for convolutional codes and an asympotoically optimum decoding algorithim. In *IEEE Transactions on Information Theory*, 13, pages 260–269. IEEE, 1967.

[Wer90] P. Werbos. Backpropagation through time: What it does and how to do it. In *Proceedings of the IEEE*, volume 78, page 1550 1560, 1990.

[WYM13] David Wemhoener, Ismet Zeki Yalniz, and R. Manmatha. Creating an improved version using noisy OCR from multiple editions. In *Proceedings of the 2013 12th International Conference on Document Analysis and Recognition*, ICDAR '13, pages 160–164, Washington, DC, USA, 2013. IEEE Computer Society.

[WZ95] R. J. Williams and D. Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. In *Back-propagation: Theory, Architectures and Application*, page 433486, 1995.

[YM11] Ismet Zeki Yalniz and Raghavan Manmatha. A fast alignment scheme for automatic OCR evaluation of books. In *ICDAR'11*, pages 754–758, 2011.

[ZD95] Justin Zobel and Philip Dart. Finding approximate matches in large lexicons, 1995.

[ZFB$^+$14] Francisco Zamora-Martínez, Volkmar Frinken, Salvador España Boquera, María José Castro Bleda, Andreas Fischer, and Horst Bunke. Neural network language models for off-line handwriting recognition. *Pattern Recognition*, 47(4):1642–1652, 2014.

# Curriculum Vitae

## Education

| | |
|---|---|
| 1997–1998 | Baccalaureate at Al-Resala Secondary School, Baghdad, Iraq. |
| 1998–2002 | Bachelor Degree in Computer Science (B.Sc.), University of Mustansiriya, Faculty of Science, Department of Computer Science, Baghdad, Iraq. Thesis: The Computer and the Arabic Language: Computerized Application in Morphology, Syntax and Semantic. |
| 2002–2004 | Master of Science Degree (M.Sc.) in Information Technology, Iraqi Commission for Computer and Information, Baghdad, Iraq. Thesis: Image Reconstruction using Wavelet Network. |

## Academic and Professional Experience

| | |
|---|---|
| 2014–present | Researcher in the Competence Center Multimedia Analysis and Data Mining (MADM) at DFKI GmbH, Kaiserslautern. |
| 2009–2014 | Researcher, and teaching assistant in the Image Understanding and Pattern Recognition Research Group (IUPR), Department of Computer Science, Technical University of Kaiserslautern. |