Robust Information Extraction From Unstructured Documents

Dissertation

zur

Erlangung des Doktorgrades (Dr. rer. nat.)

der

Mathematisch-Naturwissenschaftlichen Fakultät

der

Rheinischen Friedrich-Wilhelms-Universität Bonn

vorgelegt von

Marcin Namysł

aus

Ostrów Wielkopolski, Polen

Bonn, 2022

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät
der Rheinischen Friedrich-Wilhelms-Universität Bonn

Erster Gutachter:	Prof. Dr. Sven Behnke
Zweiter Gutachter:	Prof. Dr. Christian Bauckhage
Tag der Promotion:	7. Dezember 2022
Erscheinungsjahr:	2023

Dedication

This dissertation is dedicated to three beloved people who mean so much to me. First and foremost, to my wife Paulina Myler for her ceaseless and limitless support. Without her by my side, I would never have completed this chapter of my life. Next, to my respectful parents Barbara and Kazimierz Namysł for their understanding and encouragement during my entire Ph.D. journey.

Abstract

In computer science, robustness can be thought of as the ability of a system to handle erroneous or nonstandard input during execution. This thesis studies the robustness of the methods that extract structured information from unstructured documents containing human language texts. Unfortunately, these methods usually suffer from various problems that prevent achieving robustness to the nonstandard inputs encountered during system execution in real-world scenarios.

Throughout the thesis, the key components of the information extraction workflow are analyzed and several novel techniques and enhancements that lead to improved robustness of this process are presented. Firstly, a deep learning-based text recognition method, which can be trained almost exclusively using synthetically generated documents, and a novel data augmentation technique, which improves the accuracy of text recognition on low-quality documents, are presented. Moreover, a novel noise-aware training method that encourages neural network models to build a noise-resistant latent representation of the input is introduced. This approach is shown to improve the accuracy of sequence labeling performed on misrecognized and mistyped text. Further improvements in robustness are achieved by applying noisy language modeling to learn a meaningful representation of misrecognized and mistyped natural language tokens. Furthermore, for the restoration of structural information from documents, a holistic table extraction system is presented. It exhibits high recognition accuracy in a scenario, where raw documents are used as input and the target information is contained in tables. Finally, this thesis introduces a novel evaluation method of the table recognition process that works in a scenario, where the exact location of table objects on a page is not available in the ground-truth annotations.

Experimental results are presented on optical character recognition, named entity recognition, part-of-speech tagging, syntactic chunking, table recognition, and interpretation, demonstrating the advantages and the utility of the presented approaches. Moreover, the code and the resources from most of the experiments have been made publicly available to facilitate future research on improving the robustness of information extraction systems.

Acknowledgements

I would like to express my deepest gratitude to my advisor Prof. Dr. Sven Behnke for giving me the opportunity to pursue my research interest and develop the ideas that resulted in the approaches presented in this thesis. Moreover, I would like to thank Dr. Joachim Köhler for his encouragement and countless support during my Ph.D. research and my work at Fraunhofer IAIS. Furthermore, I would like to thank all co-authors and anonymous reviewers of the papers included in this thesis as well as the members of my doctoral committee: Prof. Dr. Christian Bauckhage, Prof. Dr. Thomas Schultz, and Prof. Dr. Andrea Schnepf.

My special thanks go to Alexander Esser, who helped me practice my conference presentations and proofread several chapters from the draft of this thesis, providing many insightful comments that helped improve the final content. Moreover, I would like to thank Tim Adams for a great collaboration during the preparation of our joint paper on benchmarking table recognition systems. Furthermore, I would like to thank Dr. Michael Gref for the exchange of experiences on the formal requirements of the application for admission to the doctoral procedure.

Several publications included in this thesis were supported by the following grants: (1) The Fraunhofer Internal Programs under the *Human Brain Pharmacome* initiative (Grant No. 836 885). (2) The German Federal Ministry of Education and Research-funded program *KMU-innovativ* in the project *DeepER*. The detailed information about the funding programs can be found in the acknowledgments sections of the corresponding publications.

Contents

1 Introduction

1.1 Background

In this thesis, we study the robustness of information extraction (IE) systems that process either born-digital Portable Document Format (PDF) files with embedded text or digitized documents in image format. In the following, the definition of the robustness concept is presented and explained intuitively. Subsequently, the task of performing IE from documents is described in detail and its typical workflow is presented.

1.1.1 Definition of Robustness

Robustness is a fundamental property of both biological organisms and sophisticated engineering systems. In biology, robust traits often result from evolutionary adaptations and are developed to ensure that specific functions are maintained despite external or internal perturbations (Kitano, 2004). Similarly, in computer science, robustness can be thought of as the ability of a system to cope with erroneous or nonstandard input during execution. IEEE Standard Glossary of Software Engineering Terminology defines robustness as: "The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions" (IEEE, 1990, p. 64).

In both biology and computer science, there are intrinsic trade-offs between robustness against certain perturbations, performance under normal circumstances, and fragility elsewhere (Kitano, 2004). Diseases expose the fragility of biological organisms and often require systematic countermeasures to reestablish control and balance the system's dynamics. Similarly, adversarial examples — input perturbations that cause machine learning (ML) models to make incorrect predictions — reveal the vulnerability of neural network models (Goodfellow et al., 2015). Moreover, the factors like the discrepancy between the data used to develop the system and the data encountered during execution in a real-world scenario often lead to decreased performance (Müller et al., 2020).

1.1.2 Information Extraction

The goal of the IE task is to extract structured information from unstructured documents containing human language texts by means of natural language processing (NLP). By creating a structured view of the information present in the input documents, IE algorithms help us to cope with the enormous amount of data that otherwise could not be processed manually.

Figure 1.1 presents the flow of the IE process. Preprocessing is first performed to clean and prepare the raw input document for subsequent processing. When documents in image format are given as input, skew angle correction (Le et al., 1994), binarization (Sauvola et al., 1997), and text line detection (Diem et al., 2013) should be performed prior to the digitization process, i.e, optical character recognition (OCR) that converts the document image into the machine-readable format. Concerning the born-digital documents, preprocessing could involve tokenization (Webster and Kit, 1992), stemming (Jivani et al., 2011), or spelling correction (Brill and Moore, 2000).

IE typically includes entity-centric tasks like named entity recognition (NER), relation extraction (Q. Zhang et al., 2017), coreference resolution (Lee et al., 2017), and knowledge base population (Ji and Grishman, 2011). Noteworthy is the NER task, whose goal is to locate all named entities in unstructured text and to classify them into predefined categories, e.g., person names, organizations, and locations (Tjong Kim Sang and De Meulder, 2003). IE could also be used to restore the structure of the information in the input document and involves tasks such as table extraction (Tengli et al., 2004), mathematical formula recognition (Lavirotte and Pottier, 1997), and other image-to-markup generation tasks (Y. Deng et al., 2017).

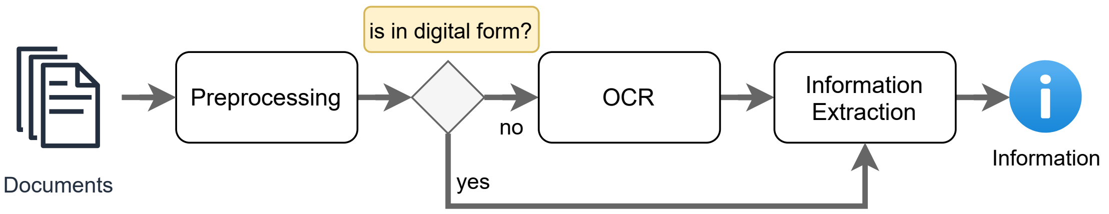


Figure 1.1: Flow of the IE process. Preprocessing is performed to clean and prepare the raw input documents for subsequent analysis. OCR is used to digitize the input document images, i.e., to convert them into a machine-readable format. OCR could be skipped if the input document is already available in a digital form. Subsequently, IE is employed to extract structured information (e.g., entities, tables) from unstructured document text.

1.2 Motivation

This thesis is primarily concerned with the robustness of the OCR, NER, and table extraction tasks. These tasks play an important role in the IE pipeline for the following reasons:

- OCR is an essential component of any IE process that works with document images that were, e.g., scanned or captured with a camera.
- NER accuracy is critical to any entity-centric downstream NLP task such as coreference resolution, which finds all expressions that refer to the same entity in a document.
- Table extraction restores the structure of information in a document, which facilitates further information and knowledge extraction activities.

Unfortunately, these tasks usually suffer from various problems that prevent achieving robustness to the nonstandard inputs encountered during system execution in real-world scenarios:

- Due to the heterogeneity of documents, the degradation of the printed material, and the distortions induced by the image acquisition process, OCR often suffers from recognition errors (Packer et al., 2010).
- NLP systems are generally trained using standard, error-free textual input but they are also employed to process user-generated text or consume the output of prior OCR or automatic speech recognition (ASR) processes. These systems thus perform poorly when the nonstandard, i.e., misrecognized or misspelled text is given as input (Miller et al., 2000).
- NLP models also suffer from the data sparsity problem of natural language text, i.e., most of the misrecognized or mistyped words are unobserved during training and therefore poorly represented by the standard NLP models (Allison et al., 2006).
- Table extraction systems are often designed to handle only one input format: either PDF files with embedded text or documents in image format (cf. L. Gao et al. (2019) and Göbel et al. (2013)). Moreover, the approaches that perform the complete table extraction process are scarce.
- Although many table extraction systems already perform well on tables with a simple layout, they often struggle with tables that have a complex layout as training data for such tables is very limited (Chi et al., 2019). Moreover, many systems perform poorly on the examples from an out-of-domain data distribution.

In this thesis, we take a step toward developing a solution to these problems, which are of high importance to the research community.

1.3 Contributions

The key contributions presented in this thesis are summarized as follows:

Robust neural OCR model and a novel data augmentation method

Chapter 2 presents a segmentation-free, deep learning-based OCR model that was trained almost exclusively using synthetically generated documents. The presented approach does not require training data that is annotated at a fine-grained level. Therefore, it can be trained using the text annotated at the level of text lines, which saves a considerable manual effort, previously required for producing the character- or word-level ground truth segmentation and transcription. Moreover, the impact of various geometrical distortions and pixel-level perturbations on the text recognition performance is analyzed and a novel data augmentation technique — *alpha compositing with background texture images* — is proposed. The performed experiments show that: (1) The proposed data augmentation method improves the robustness of neural OCR methods for distorted images and (2) The developed text recognition system outperforms established commercial and open-source OCR engines in terms of accuracy on challenging benchmark data sets consisting of both real and synthetically rendered documents.

Noise-aware training (NAT) method

In Chapter 3, we examine neural network models for sequence labeling by feeding the text that contains naturally occurring adversarial examples — OCR errors and misspellings. Firstly, two variants of a confusion matrix-based error model are defined. The *vanilla* variant represents plausible typographic noise that mimics the naturally occurring errors using a uniform distribution over the set of edit operations at the character level. In contrast, the *empirical* noise model employs the error distribution encountered in real-world scenarios, which is estimated by aligning pairs of error-free and erroneous sentences. Secondly, a novel noise induction procedure that simulates misrecognized or misspelled sentences is proposed. Thirdly, the NAT method is introduced. It employs two auxiliary training objectives: *data augmentation* and *stability training*, which use both the original, error-free input and its synthetically distorted variants to improve the robustness of the sequence labeling models to erroneous text. The effectiveness of the presented approach was demonstrated by comparing the accuracy of state-of-the-art baseline models trained using either the proposed NAT method or the standard training objective for the sequence labeling task on text containing real-world OCR errors and misspellings.

Empirical error generation and noisy language modeling (NLM) method

Chapter 4 presents an improved version of the NAT framework. Firstly, the confusion matrix-based noising process is replaced with a learnable error generation method that employs a sequence-to-sequence (Sutskever et al., 2014) model trained to translate from error-free to erroneous text. To train the error generation model, an unsupervised parallel training data generation method is developed, which directly utilizes an OCR engine and accurately simulates naturally occurring OCR noise distribution. The presented empirical error induction method improves the accuracy of noisy neural sequence labeling compared to the baseline error model from prior work. Secondly, to overcome the *data sparsity* issue, the NLM method is proposed, which estimates the embeddings employed by the sequence labeling model using noisy text produced by the unsupervised parallel data generation process. The NLM embeddings further improve the accuracy of models for two sequence labeling tasks in a scenario where the input text contains OCR or human-generated errors.

Flexible table recognition and semantic interpretation method

In Chapter 5, a holistic approach is introduced that performs recognition and semantic interpretation of tables contained in unstructured documents that are either in image or PDF format. For table recognition, a hybrid method is implemented that combines a deep learning-based table detection module with heuristics for table structure recognition (TSR). The proposed algorithms recognize tables with various formats such as fully bordered, partially bordered, and borderless tables. Moreover, the basic formulation of the table recognition task is complemented by including a table interpretation module. To this end, a general formulation of the table interpretation task as a *maximum weight matching* (Edmonds, 1965) on a corresponding graph is provided and a rule-based table interpretation method is proposed. This method leverages regular expressions (RegEx) (Kleene, 1951) and an approximate string-matching algorithm to compute semantic similarities between table cells and predefined semantic concepts. The proposed table recognition method was evaluated on two challenging benchmarks and achieved results on par with the state-of-the-art approaches in this domain. Moreover, an issue in the official repository of a recent competition on table recognition was corrected and the results of the proposed and the baseline method obtained using the rectified evaluation script are presented. Finally, the proposed holistic table extraction system exhibited high recognition accuracy in an end-to-end IE scenario, where raw documents were used as input and target information was contained in tables.

A novel evaluation method of the complete table recognition (CTR) task

In Chapter 6, we examine the accuracy of several general-purpose table recognition systems that exhibit state-of-the-art results on widely used table recognition benchmarks in a scenario, where the input documents come from the biomedical domain (Adams and Namysl, 2021). As the employed benchmarking data set does not provide information about the exact location of table objects on a page, a novel method that evaluates the CTR process was designed. It performs table matching and structure comparison between the ground-truth and the recognized tables using exclusively the information about the structure of the tables and not their exact location in a document. The results obtained by each baseline method on the employed benchmarking data set are discussed, as well as the factors that can impact the performance of table recognition systems.

1.4 Publications

Parts of this thesis have been published in conference proceedings and journals. The most relevant publications[1] are presented below in chronological order:

Pub. (1) M. Namysl and I. Konya (2019). "Efficient, lexicon-free OCR using deep learning." In: *International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, pp. 295–301. DOI: 10.1109/ICDAR.2019.00055

Pub. (2) M. Namysl, S. Behnke, J. Köhler (2020). "NAT: Noise-aware training for robust neural sequence labeling." In: *Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 1501–1517. DOI: 10.18653/v1/2020.acl-main.138

[1] In Appendix C, the full text of each publication is reprinted with the permission of the copyright holders.

Pub. (3) M. Namysl, S. Behnke, J. Köhler (2021). "Empirical error modeling improves robustness of noisy neural sequence labeling." In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP*. Association for Computational Linguistics, pp. 314–329. DOI: 10.18653/v1/2021.findings-acl.27

Pub. (4) T. Adams, M. Namysl, A. T. Kodamullil, S. Behnke, and M. Jacobs (2021). "Benchmarking table recognition performance on biomedical literature on neurological disorders." In: *Bioinformatics* 38.6, pp. 1624–1630. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btab843

Pub. (5) M. Namysl, A. Esser, S. Behnke, J. Köhler (2022). "Flexible table recognition and semantic interpretation system." In: *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications.* Vol. 4: VISAPP. INSTICC. SciTePress, pp. 27–37. ISBN: 978-989-758-555-5. DOI: 10.5220/0010767600003124

It is worth noting that the publication that is marked as Pub. (5) in the list above was nominated for the *Best Industrial Paper Award* at the 17th International Conference on Computer Vision Theory and Applications.[2]

In addition to the key publications that constitute the core part of this thesis, the author also contributed to the following publications that are closely related to the topics presented in this thesis and were written during the time in which the presented research has been conducted:

- K. D. Dhole, V. Gangal, S. Gehrmann, A. Gupta, Z. Li, et al. (2021). *NL-Augmenter: A framework for task-sensitive natural language augmentation.* DOI: 10.48550/ARXIV.2112.02721
- V. Lage-Rupprecht, B. Schultz, J. Dick, M. Namysl, A. Zaliani, et al. (2022). "A hybrid approach unveils drug repurposing candidates targeting an Alzheimer pathophysiology mechanism." In: *Patterns* 3.3, p. 100433. ISSN: 2666-3899. DOI: 10.1016/j.patter.2021.100433

1.5 Outline

This thesis consists of seven chapters. Chapters 2-6 present the scientific contributions of the thesis (see Section 1.3). Each of them starts with the introduction to the topic of interest, and problem formulation followed by the details of

[2] https://visapp.scitevents.org/?y=2022

the proposed methods, the experimental evaluation, and the discussion of the advantages and limitations of the presented approach. Moreover, a review of the related approaches in the field is also presented. Finally, Chapter 7 concludes this thesis by discussing the accomplished scientific results and possible future work directions.

1.6 Open-Source Software

To facilitate future research on improving the robustness of IE systems, most of the described methods and proposed data sets were made publicly available, including:

- The complete source code of the NAT framework that was introduced in Namysl et al. (2020).[3]
- The complete source code of the improved NAT framework including resources used in the experiments and the pretrained NLM embeddings presented in Namysl et al. (2021).[4]
- The source code of the evaluation method of the CTR process, which was presented in Adams et al. (2021).[5]
- The table recognition benchmark (Adams and Namysl, 2021) that was introduced in Adams et al. (2021).[6]
- The source code of the evaluation method of the table interpretation task and the resources used in the table interpretation experiment presented in Namysl et al. (2022).[7]
- The resources from the table recognition experiment presented in Section 5.7, including the evaluation scripts and the output files produced by the proposed and the baseline approach.[8]

Moreover, the author also contributed to the development of the *NL-Augmenter* framework[9] (Dhole et al., 2021) — a collaborative effort intended to add reusable transformations of NLP data sets that could be used to generate additional training data or to test model robustness.

[3] `https://github.com/mnamysl/nat-acl2020`
[4] `https://github.com/mnamysl/nat-acl2021`
[5] `https://github.com/mnamysl/benchmarking_table_recogn`
[6] `https://zenodo.org/record/5549977`
[7] `https://github.com/mnamysl/table-interpretation`
[8] `https://github.com/mnamysl/tabrec-sncs`
[9] `https://github.com/GEM-benchmark/NL-Augmenter`

2 Robust Neural OCR Engine

Preface

This chapter is adapted from Namysl and Konya (2019)[1], previously published by IEEE and presented at the 15th International Conference on Document Analysis and Recognition (ICDAR 2019)[2].

Statement of Personal Contribution

The author of this thesis substantially contributed to all aspects of the previous publication (Namysl and Konya, 2019), including the conception, design, and implementation of the proposed methods, the collection, generation, and annotation of the data for training and evaluation of the proposed approach, conducting the experimental evaluation, the analysis and interpretation of the experimental results, drafting the manuscript, as well as the revision, proofreading, and final approval of the version to be published.

The content presented in this chapter, unless otherwise stated, is the contribution of the author of this thesis.

Unpublished Content

Compared to the previous publication, additional, unpublished content is included in this chapter. In particular, Section 2.2 that describes the preliminaries and Section 2.6.4 that discusses the employed decoding method on a qualitative example were additionally presented.

[1] ©2019 IEEE. Reprinted in Appendix C.1 with permission, from M. Namysl and I. Konya (2019). "Efficient, lexicon-free OCR using deep learning." In: *International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, pp. 295–301. DOI: `10.1109/ICDAR.2019.00055`
[2] `http://icdar2019.org`

Abstract

OCR remains a challenging problem when text occurs in unconstrained environments, like natural scenes, due to geometrical distortions, complex backgrounds, and diverse fonts. In this chapter, a segmentation-free text recognition system that combines deep learning-based methods and rich data augmentation techniques is presented. It utilizes synthetic training data that was generated by rendering images, which contain sentences sampled from a large text corpus, using diverse fonts. To simulate text occurring in complex natural scenes, rendered images are augmented with geometric distortions and with a proposed data augmentation technique — *alpha compositing with background textures*. The employed architecture uses a convolutional neural network (CNN) encoder to extract features from text images. To model the interactions between the input elements, both CNN and recurrent neural network (RNN) models are used. The proposed method surpasses the accuracy of leading commercial and open-source OCR engines on distorted text examples.

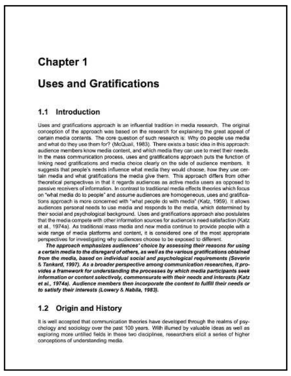
2.1 Introduction

OCR is one of the most widely studied problems in the field of pattern recognition and computer vision. OCR is an important component of the digitization process, which involves converting analog document sources to their electronic versions for further processing, analysis, or archiving. It is not limited to printed text but also includes handwritten documents, as well as natural scene text (Figure 2.1).

Typically, the OCR task involves two components: a text detection and a text recognition module.[3] The goal of the text detection component is to localize all text blocks, i.e., characters, words, lines, or paragraphs, within an image. Subsequently, the text recognition module aims to transcribe the text depicted in each textual region into natural language tokens.

The accuracy of various OCR methods has recently greatly improved due to advances in deep learning (He et al., 2016; Hu et al., 2018; Ioffe and Szegedy, 2015). Moreover, many current open-source and commercial products reach a high recognition accuracy and good throughput for run-of-the-mill printed document images. While the recent progress in this field has led the research community to regard OCR as a largely solved problem, this work shows that

[3] Long et al. (2021) present a thorough review of recent scene text detection and recognition approaches.



Chapter 1

Uses and Gratifications

1.1 Introduction

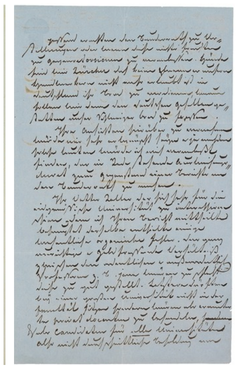
Uses and gratifications approach is an influential tradition in media research. The original conception of the approach was based on the research for explaining the great appeal of certain media contents. The core question of such research is: Why do people use media and what do they use them for? (McQuail, 1983). There exists a basic idea in this approach: audience members know media content, and which media they can use to meet their needs. In the mass communication process, uses and gratifications approach puts the function of linking need gratifications and media choice clearly on the side of audience members. It suggests that people's needs influence what media they would choose, how they use certain media and what gratifications the media give them. This approach differs from other theoretical perspectives in that it regards audiences as active media users as opposed to passive receivers of information. In contrast to traditional media effects theories which focus on "what media do to people" and assume audiences are homogeneous, uses and gratifications approach is more concerned with "what people do with media" (Katz, 1959). It allows audiences personal needs to use media and responds to the media, which determined by their social and psychological background. Uses and gratifications approach also postulates that the media compete with other information sources for audience's need satisfaction (Katz et al., 1974a). As traditional mass media and new media continue to provide people with a wide range of media platforms and content, it is considered one of the most appropriate perspectives for investigating why audiences choose to be exposed to different.

The approach emphasizes audiences' choice by assessing their reasons for using a certain media to the disregard of others, as well as the various gratifications obtained from the media, based on individual social and psychological requirements (Severin & Tankard, 1997). As a broader perspective among communication researches, it provides a framework for understanding the processes by which media participants seek information or content selectively, commensurate with their needs and interests (Katz et al., 1974a). Audience members then incorporate the content to fulfill their needs or to satisfy their interests (Lowery & Nabila, 1983).

1.2 Origin and History

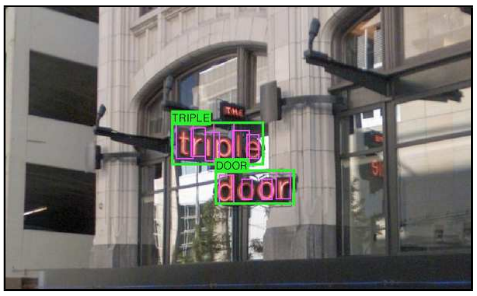
It is well accepted that communication theories have developed through the realms of psychology and sociology over the past 100 years. With illumed by valuable ideas as well as exploring more untilled fields in these two disciplines, researchers elicit a series of higher conceptions of understanding media.

(a) Printed text example (©2015 IEEE)



[illegible]

(b) Handwritten text example (©2017 IEEE)



(c) Scene text example (©2011 IEEE)

Figure 2.1: Examples of (a) printed, (b) handwritten, and (c) scene text. The images were reprinted from Nayef et al. (2015), Sánchez et al. (2017), and K. Wang et al. (2011), respectively.

even the most successful and widespread OCR solutions are neither able to robustly handle large font varieties nor distorted texts, potentially superimposed on complex backgrounds. Such unconstrained environments for digital documents have already become predominant, due to the wide availability of mobile phones and various specialized video recording devices (Nayef et al., 2015). Moreover, existing methods seem to be suboptimized for some commonly used subsets of fonts. They exhibit high susceptibility to foreground and background variations and considerable dependence on external language models (Smith, 2011).

In contrast to popular OCR engines, methods used in scene text recognition (STR) (Bušta et al., 2017; Lyu et al., 2018) exploit computationally expensive network models, aiming to achieve the best possible recognition rates on popular benchmarks. Such methods are tuned to deal with significantly smaller amounts of text per image and are often constrained to predefined lexicons. Commonly used evaluation protocols substantially limit the diversity of symbols to be recognized, e.g., by ignoring all non-alphanumeric characters, or by neglecting case sensitivity (K. Wang et al., 2011). Hence, models designed for scene text are generally inadequate for performing OCR of printed documents, where high throughput and support for great varieties of symbols are essential.

The method presented in this chapter addresses the general text recognition problem and tries to overcome the limitations of both printed OCR and STR systems. To this end, a fast and robust multiple-font OCR engine was developed. It recognizes 132 different character classes and employs deep learning-based models that are trained almost exclusively using synthetically generated documents. The proposed method requires a much lower data labeling effort, making the resulting framework more readily extensible for new languages and scripts. Moreover, in Section 2.4.3, a novel data augmentation technique is proposed to improve the robustness of neural models for text recognition.

The presented approach was evaluated using large and challenging data sets that consist of both real and synthetically rendered documents. The comparison with leading commercial and established open-source engines shows that the proposed solution obtains significantly better recognition accuracy with comparable execution time.

2.2 Preliminaries

In this section, the RNN and the CNN architectures employed by the proposed text recognition method, as well as by several related approaches reviewed in Section 2.3, are briefly introduced. This part can be omitted if the reader is already familiar with the aforementioned neural network architectures and their popular variants.

2.2.1 Recurrent Neural Networks

An RNN is a class of artificial neural networks designed to model interactions between consecutive input elements. It maintains its internal state to better process sequential data. RNNs can be trained with iterative gradient descent algorithms like backpropagation through time (Werbos, 1990).

An RNN computes its hidden states h_t and output vectors y_t as follows (Graves et al., 2013):

$$h_t = \mathcal{H}(W_{xh}x_t + W_{hh}h_{t-1} + b_h), \tag{2.1}$$

$$y_t = W_{hy}h_t + b_y, \tag{2.2}$$

where x is the input vector and t denotes the time step that iterates from 1 to the maximum number of steps T. Moreover, W_{xh}, W_{hh}, and W_{hy} represent the input-hidden, hidden-hidden, and hidden-output weight matrices, respectively. Furthermore, b_h and b_y are the corresponding bias vectors. $\mathcal{H}$ is the hidden layer function modeled as an element-wise application of a *sigmoid* function (Graves et al., 2013).

A long short-term memory (LSTM) network (Hochreiter and Schmidhuber, 1997) is a special variant of the RNN consisting of LSTM units that implement a more complex hidden layer function $\mathcal{H}$. LSTM networks prevent backpropagated errors from vanishing or exploding (Pascanu et al., 2013), and therefore can memorize particular events for a longer period of time than the standard RNN.

A basic LSTM unit consists of a memory cell and three multiplicative gates, i.e., input, forget, and output gates. A more elaborated extension is the *peephole* LSTM unit (Gers and Schmidhuber, 2000), where the multiplicative gates compute their activations at the current time step using additionally the activation of the memory cell from the previous time step (Figure 2.2).

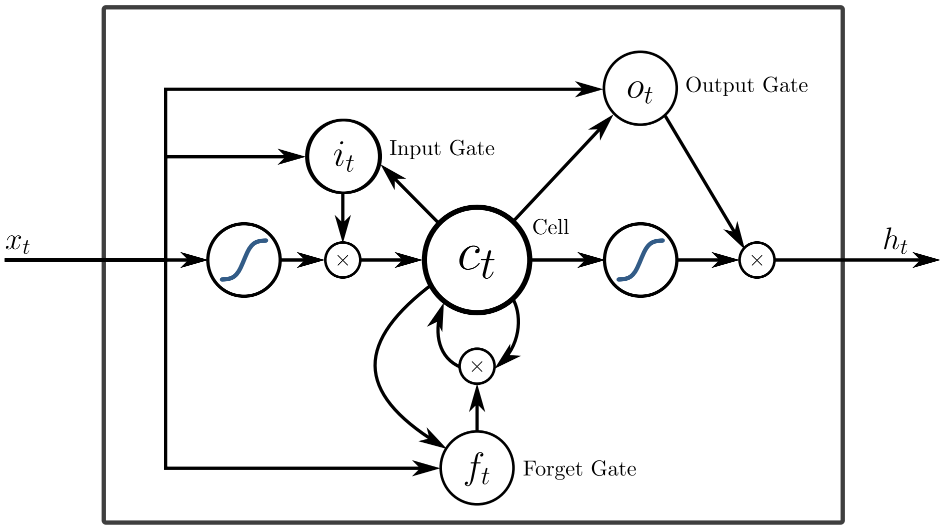


Figure 2.2: Illustration of a peephole LSTM unit. The memory cell is denoted as c_t. The sigmoid function is represented by an S-shaped curve. ©2013 IEEE. Reprinted from Graves et al. (2013).

A gated recurrent unit (GRU) (Cho et al., 2014) is another variant of the RNN unit with an alternative gating mechanism. GRUs have only the update and reset gates and therefore fewer parameters and simpler structure than the LSTM units.

The RNN architecture was further extended to better exploit the information from past and future states and to be able to process multidimensional data. Bidirectional recurrent neural networks (BRNNs) (Schuster and Paliwal, 1997) consist of two hidden RNN layers traversing the input sequence in opposite spatial directions (i.e., left-to-right and right-to-left directions) that are connected to a single output layer, as illustrated in Figure 2.3. The BRNN computes the sequences of forward and backward hidden state vectors ($\overrightarrow{h}$ and $\overleftarrow{h}$, respectively), and the output vector as follows (Graves et al., 2013):

$$y_t = W_{\overrightarrow{h}y}\overrightarrow{h}_t + W_{\overleftarrow{h}y}\overleftarrow{h}_t + b_y. \tag{2.3}$$

Multidimensional recurrent neural networks (MDRNNs) (Graves et al., 2007) further generalize standard RNNs by providing recurrent connections along all spatiotemporal dimensions, making them robust to local distortions along any combination of the respective input dimensions.

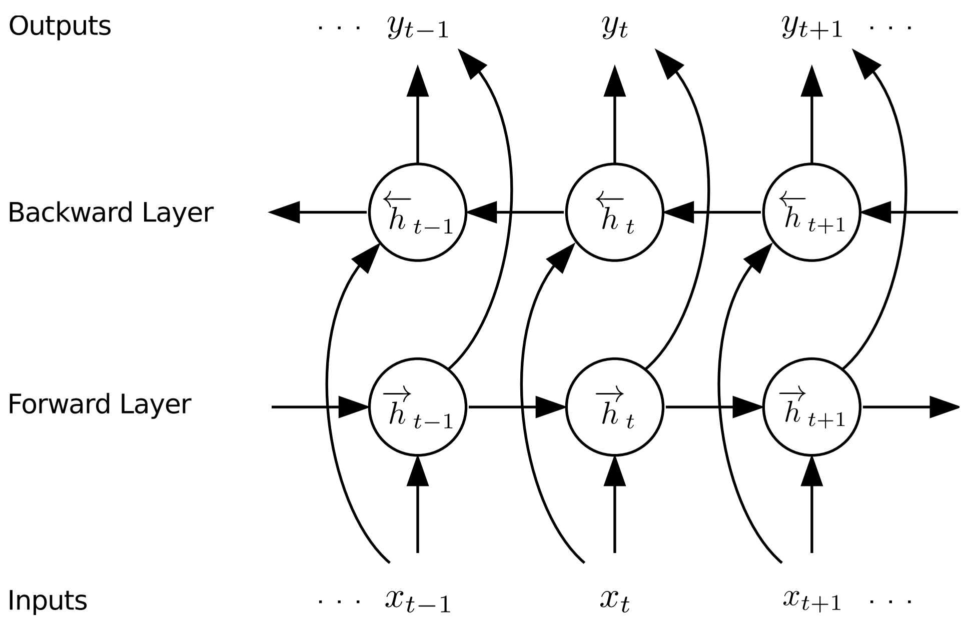


Figure 2.3: Illustration of a bidirectional RNN. x, y, and h are the input, output, and hidden state vectors, respectively, and t denotes the time step. Arrows indicate the direction in which the input sequence is traversed. ©2013 IEEE. Reprinted from Graves et al. (2013).

2.2.2 Convolutional Neural Networks

A CNN (Lecun et al., 1998) is a class of feed-forward neural networks commonly used to extract features from speech, images, and videos (Chauhan et al., 2018; Huang et al., 2014; Xu et al., 2015). These features can be then used, e.g., for object classification and detection (Krizhevsky et al., 2012).

CNNs use a concept of the receptive field, where each neuron of a hidden layer receives inputs only from a local region in the previous layer, which greatly reduces the number of weights that need to be learned (Le and Borji, 2017; Luo et al., 2016). Such connectivity patterns between neurons are biologically inspired by the organization of the animal visual cortex (Hubel and Wiesel, 1962).

Each convolutional layer applies a cross-correlation operation with a learnable two-dimensional kernel matrix to its inputs and passes the results to the next layer (Figure 2.4). This allows the CNN to be deeper in terms of the number of layers than the corresponding feed-forward network with the same number of parameters (Simonyan and Zisserman, 2014; Szegedy et al., 2015). Convolutional filters learned this way successfully replaced traditionally used, hand-crafted features in many computer vision applications (Coşkun et al., 2017; Kagaya et al., 2014).

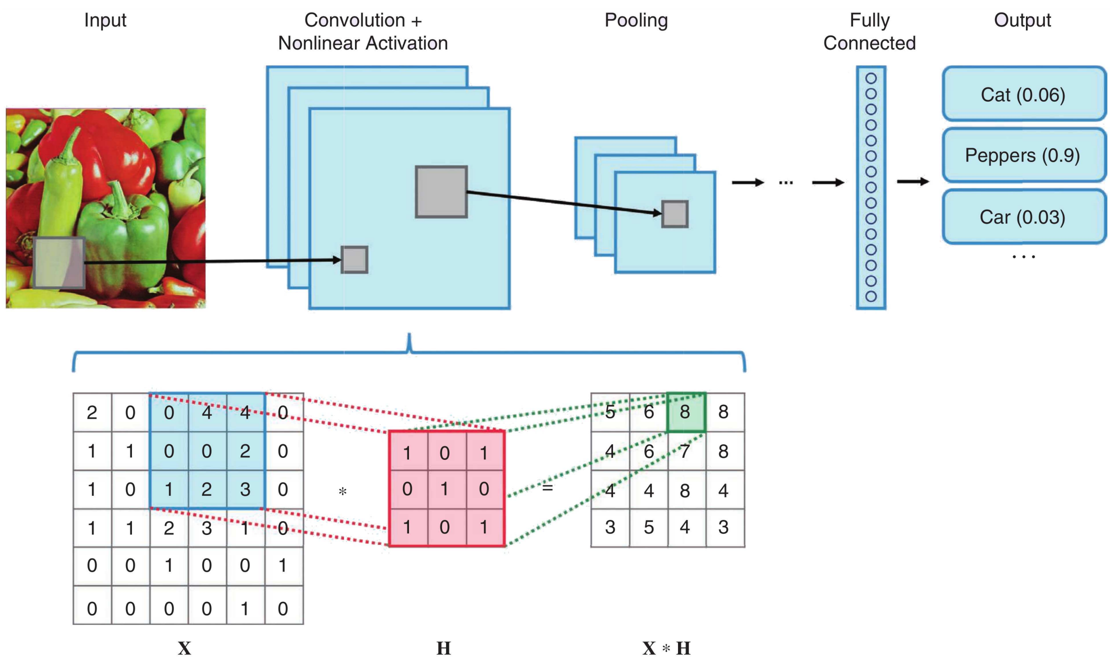


Figure 2.4: Illustration of a CNN. Typically, it consists of several convolutional layers with a nonlinear activation function interleaved with pooling layers, which reduce the dimensions of the feature maps. The feature maps from the last layer are fed to a fully connected neural network with a *softmax* function (Bridle, 1990) to get the output predictions. A convolutional layer applies the cross-correlation (denoted as $*$) operation to the input X with a kernel H, as illustrated in the lower part of the figure. ©2020 IEEE. Adapted from Cheung et al. (2020).

2.3 Related Work

In this section, a detailed review of related approaches for printed, handwritten, and scene text recognition is presented. These can be broadly categorized into *segmentation-based* and *segmentation-free* methods, which are detailed in Sections 2.3.1 and 2.3.2, respectively.

2.3.1 Segmentation-Based Text Recognition Methods

Segmentation-based methods recognize individual character hypotheses, explicitly or implicitly generated by a character segmentation method (Breuel, 2008; Jacobs et al., 2005). The output is a recognition lattice containing various segmentation and recognition alternatives weighted by the classifier. The lattice is then decoded, e.g., via a greedy or beam search method. Moreover, the decoding process may also make use of an external language model or allow the incorporation of certain lexicon constraints.

The PhotoOCR system for text extraction from smartphone imagery proposed by Bissacco et al. (2013) is a representative example of a segmentation-based OCR method. In this method, the segmentation points between characters are detected using a binary logistic sliding window classifier trained on a combined histogram of oriented gradients (HOG) and weighted direction code histogram feature vector. They used a deep fully connected neural network model trained on extracted HOG features for character classification and incorporated a character-level language model into the score function to facilitate resolving ambiguities between similarly looking characters.

The accuracy of segmentation-based methods heavily suffers from segmentation errors and the lack of context information wider than a single character candidate image during classification. Improper incorporation of an external language model or lexicon constraints can significantly degrade the accuracy of the complete OCR system (Smith, 2011). While offering high flexibility in the choice of segmentation, classification, and decoding methods, segmentation-based approaches require a similarly high effort to tune optimally for specific application scenarios. Moreover, the precise weighting of all involved hypotheses must be recomputed from scratch as soon as one component is updated (e.g., the language model), whereas the process of data labeling (e.g., at the character or pixel level) usually causes high costs in terms of human annotation labor.

2.3.2 Segmentation-Free Text Recognition Methods

Segmentation-free methods eliminate the need for presegmented inputs, i.e., allow to transcribe images of entire words or text lines without prior detection of character bounding boxes. In the following, the evolution of the segmentation-free methods is presented, and the most popular and successful approaches are briefly summarized.

Text Recognition Methods Based on Hidden Markov Models

The previous line of work on segmentation-free text recognition employed Hidden Markov Models (HMMs). Motivated by the success of HMMs in speech (Gales and Young, 2007) and handwritten text recognition (HTR) (Bunke et al., 2004; Natarajan et al., 2008), HMMs were also applied to printed OCR (Rashid et al., 2012). Although HMMs avoided many difficulties of segmentation-based approaches, the drawbacks of HMMs, like assuming independent observations and being generative in nature, limit their recognition performance. Therefore, most of the recently developed segmentation-free solutions employ the RNN and CNN architectures, which are described in the following part of this section.

Text Recognition Methods Based on Recurrent Neural Networks

BRNNs were found to be well-suited for both handwritten (Graves et al., 2009) and printed text recognition (Breuel et al., 2013). Moreover, MDRNNs have gained high popularity among researchers in the field of HTR (Graves and Schmidhuber, 2008; Voigtlaender et al., 2016) because of their ability to attain state-of-the-art recognition rates.

MDRNNs are computationally much more expensive than their basic one-dimensional variant, both during training and inference. Because of this, they have been less frequently explored in the field of printed document OCR. Instead, to overcome the issue of sensitivity to stroke variations along the vertical axis, researchers have proposed different solutions. For example, Breuel et al. (2013) combined a bidirectional long short-term memory (BLSTM) network architecture with a text line normalization method for performing OCR of printed Latin and Fraktur scripts. Similarly, by normalizing the positions and baselines of letters, Reza Yousefi et al. (2015) achieved superior performance and faster convergence with a standard BLSTM network over its two-dimensional variant for Arabic HTR.

An additional advantage of segmentation-free approaches is their inherent ability to work directly on grayscale or full-color images. This increases the robustness and accuracy of text recognition, as any information loss caused by a previously mandatory binarization step can be avoided. Asad et al. (2016) applied a BLSTM network directly to original, blurred document images and were able to obtain state-of-the-art recognition results. Similarly, Yousefi et al. (2015) trained a BLSTM network directly on grayscale images of text lines for OCR of historical documents.

Hybrid Text Recognition Methods

The incorporation of CNNs allowed for further improvements in the recognition accuracy. Since CNNs can extract latent representations of input images, thus increasing robustness to local distortions, they can be successfully employed as a substitute for the MDRNN layers.

Shi et al. (2017) introduced the convolutional recurrent neural network (CRNN) architecture, which is a combination of deep CNN and LSTM models. The former are used to extract sequences of features from the input image and the latter are employed to predict a distribution over the class labels for each input frame. Finally, the connectionist temporal classification (CTC) (Graves et al., 2006) layer is utilized to translate the predictions into the final sequence of labels. CTC is a variant of the forward-backward algorithm (Rabiner, 1989) that eliminates the need to use presegmented training data, which greatly facilitates the training process. In a similar manner, Breuel (2017) proposed a model that combines CNNs and LSTMs for printed text recognition. Features extracted by CNNs are combined and fed into the LSTM network with a CTC output layer. Moreover, Puigcerver (2017) applied a model that relies only on CNNs and BLSTMs to HTR and compared it against the MDRNN model, achieving competitive results with significantly faster runtime.

Text Recognition Methods Based on Fully Convolutional Networks

A few other methods have completely departed from the use of the computationally expensive RNNs and rely purely on convolutional layers for modeling the local image context. One example of such a method is a framework for STR presented by Jaderberg et al. (2014) that performs word recognition on a whole image holistically. They examined three different classification methods, i.e., dictionary, character sequence, and bag-of-N-gram encoding, on top of a CNN-based feature extraction module. They demonstrated the superior performance of their models on several standard STR benchmarks.

In contrast, Bušta et al. (2017) used a fully convolutional network (FCN) to extract relevant features from word images and a CTC layer to transform variable-width feature vectors into conditional probability distributions over available classes. FCN follows the CNN architecture but does not employ fully connected layers and thus relies exclusively on the convolution operations. Similarly, Borisyuk et al. (2018) presented a scalable OCR system called *Rosetta*, which employs an FCN model followed by the CTC layer to extract the text depicted on input images uploaded daily by the users of a popular social media platform.

Attention-Based Text Recognition Methods

Visual attention allows humans to focus on a certain region in the visual field with higher resolution while perceiving the surrounding areas less accurately (Treue and Katzner, 2009). Bahdanau et al. (2015) proposed to incorporate the attention mechanism into the standard sequence-to-sequence architecture for the neural machine translation (NMT) task. Attention was shown to help the model dynamically focus on a particular input element while making predictions in each decoding step, which improved the accuracy of the models on long input sentences.

Inspired by these findings, the attention mechanism was increasingly used in the area of STR and HTR. Lee and Osindero (2016) presented an approach that extracts and encodes image features using recursive CNNs and then decodes them using RNNs one character at a time. They use the attention mechanism to improve the feature selection process. The approach proposed by Ghosh et al. (2017) relies on an LSTM-based visual attention model that is jointly learned with a CNN-based image encoder. Their attention-based decoder emits the output characters using a weighted combination of image features corresponding to different areas of the image. Noteworthy, they integrate an explicit language model into the decoding algorithm, achieving state-of-the-art performance on standard STR benchmarks. Similarly, Chowdhury and Vig (2018) combined a deep CNN with a recurrent encoder-decoder network with attention to perform HTR. The latter was used to transcribe the text depicted in the input image. Their system surpassed the state-of-the-art word-level accuracy on two popular HTR benchmarks.

Text Recognition Methods Based on the Transformer Architecture

Motivated by the success of the attention-based models in NLP, Vaswani et al. (2017) introduced the *Transformer* architecture, which is solely based on the attention mechanism, omitting the use of recurrent or convolutional layers. Transformer-based models were successfully applied in many NLP (Devlin et al., 2019; Raffel et al., 2020) and computer vision tasks (Dosovitskiy et al., 2021; Zhu et al., 2021).

Inspired by these results, some recent approaches employed Transformer-based network models for the OCR problem. M. Li et al. (2021) proposed *TrOCR*, a text recognition approach that leverages pretrained Transformer models for both image and text modalities. Their model outperformed the state-of-the-art approaches on both printed and handwritten text recognition tasks.

In contrast, Diaz et al. (2021) investigated the general text line recognition problem. They considered two decoder families, i.e, the Transformer- and the CTC-based models, and different encoder modules, including, BLSTM- and attention-based architectures. Interestingly, their experiments on widely used data sets for STR and HTR showed that, although not previously explored for the OCR task, the attention-based encoder coupled with the CTC decoder outperformed all other combinations of encoder and decoder components in terms of accuracy and computational complexity.

2.4 Proposed Method

In this section, the architecture of the developed text recognition approach (Section 2.4.1), the proposed synthetic document generation method (Section 2.4.2), the introduced data augmentation technique (Section 2.4.3), and the employed geometrical normalization method (Section 2.4.4) are presented.

2.4.1 System Architecture

In the following, two variants of the proposed text recognition architecture depicted in Figure 2.5 are described. The presented system builds upon the segmentation-free text recognition techniques described in Section 2.3.2.

Inspired by the CRNN (Shi et al., 2017) and *Rosetta* (Borisyuk et al., 2018) systems, either a hybrid CNN-BLSTM-based or an FCN-based model is used to perform feature extraction and sequence modeling. On top of the core model, a well-established CTC transcription layer is employed. The bottom part of the architecture consists of multiple CNN layers that extract higher-level features from an input image. Activation maps obtained by the last convolutional layer are transformed into a feature sequence with the *map-to-sequence* operation. Specifically, three-dimensional maps are sliced along their width dimension into two-dimensional maps and then each map is flattened into a vector.

In the case of the hybrid CNN-BLSTM variant, the obtained feature vectors are fed into a BLSTM network with 256 hidden units in both directions. The output sequences from both layers are concatenated and fed to a linear layer with the *softmax* activation function (Bridle, 1990) to produce a per timestep probability distribution over the set of available classes. The classes correspond to the characters that can be recognized by the model. In the case of the FCN model variant, feature sequences transformed by the map-to-sequence operation are directly fed into a linear layer, skipping the recurrent components entirely.

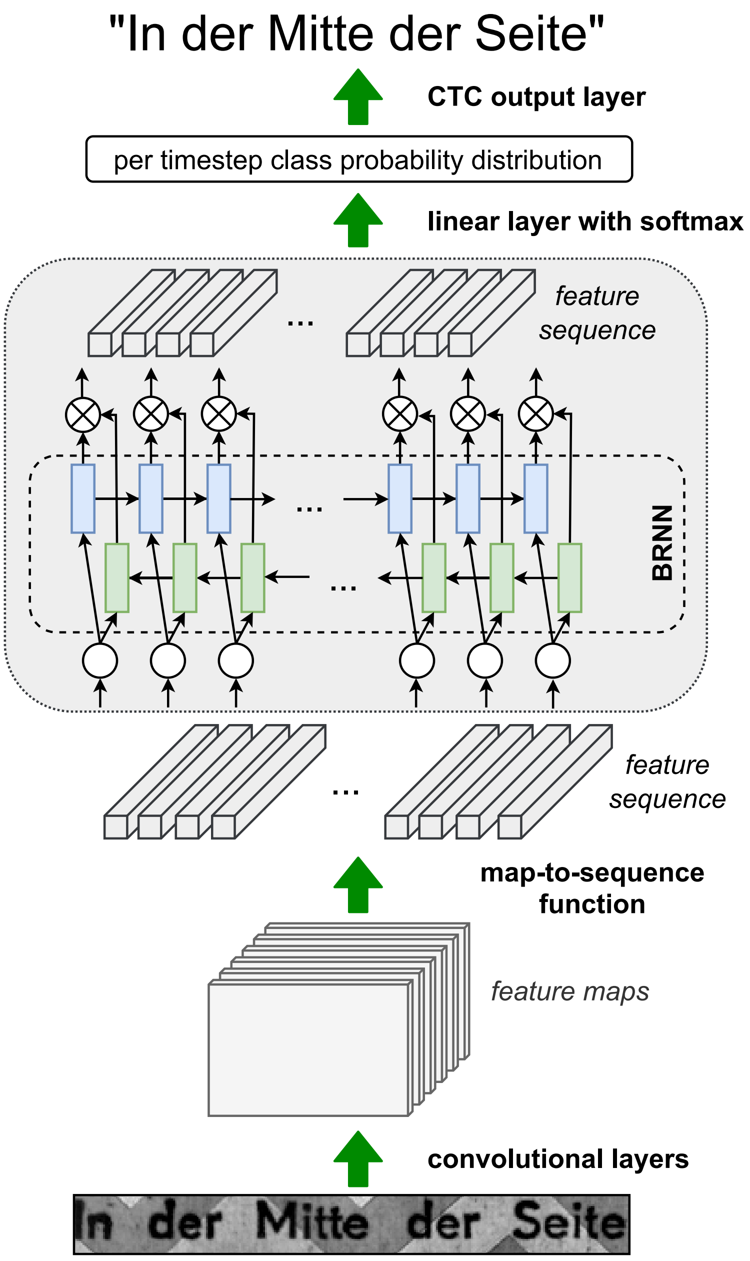


Figure 2.5: The architecture of the proposed text recognition system. An image is fed into the CNN layers that perform feature extraction. Subsequently, the obtained activation maps are transformed into feature sequences and fed into a BLSTM network, which performs modeling of the interaction between the elements of the feature vector sequence. Finally, higher-level feature sequences are transcribed into sequences of natural language tokens using a CTC output layer. The gray region indicates a recurrent block that is omitted in the case of the FCN-based model. In such a case, feature sequences extracted by deep CNNs are directly fed into the CTC component.

Finally, at training time, the CTC output layer is employed to compute a loss between the network outputs and the ground-truth transcriptions. During inference, CTC loss computation is replaced by greedy CTC decoding. Tables A.1 and A.2 in the appendix present a detailed structure of the hybrid CNN-BLSTM model and the FCN-based architecture, respectively.

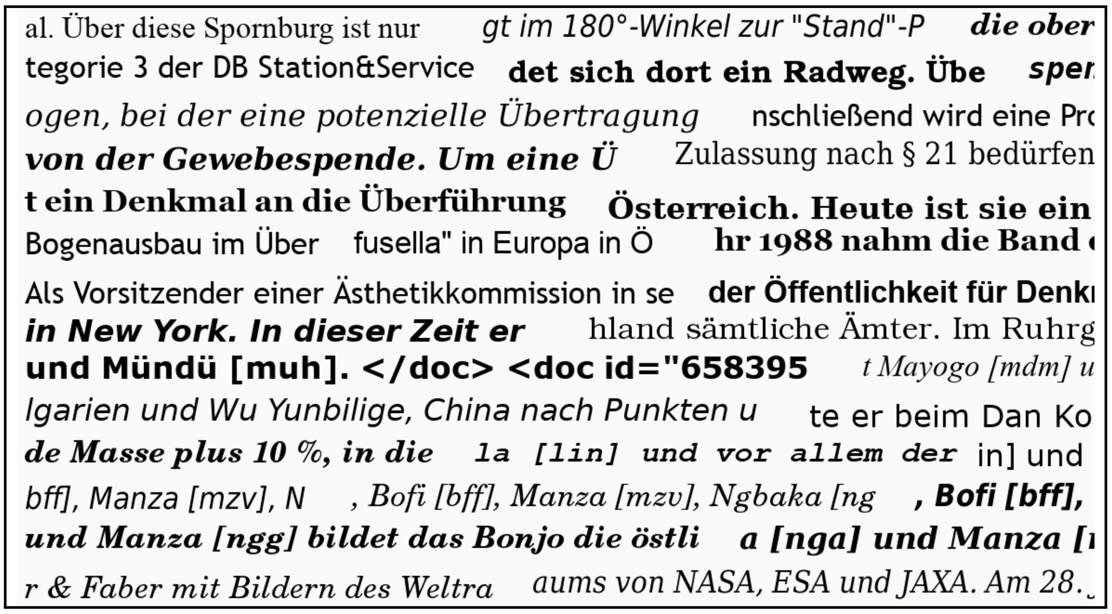
2.4.2 Synthetic Document Generation

Even though a segmentation-free text line recognition system does not need the ground-truth training data at the character or word level, the manual annotation process might still be error-prone and time-consuming. Inspired by the work of Jaderberg et al. (2014), in this work, an automatic synthetic document creation process is developed. To this end, artificial document pages with synthetically rendered lines of text are generated to be used for the training and evaluation of text line recognition systems.

Given a text corpus $\mathcal{T}$ and a set of fonts $\mathcal{F}$, the text line generation process first selects a piece of text (up to 40 characters) from $\mathcal{T}$ and renders it in an image with a font that is randomly chosen from $\mathcal{F}$. The rendered sequence of tokens and the attributes that were used for rendering (i.e., bounding boxes, baseline positions, and x-height values) are associated with the rendered image and stored in the corresponding text line metadata. Upon generating enough text line samples to fill a page image of predefined dimensions (e.g., $3{,}500 \times 5{,}000$ pixels), the image is saved on a hard disk together with the associated metadata.

Since many different Unicode symbols produce almost identical output, some characters are replaced by the most similar one that is modeled in the proposed system, e.g., *hyphen* character (`0x2010`) is replaced by *hyphen-minus* character (`0x2D`). In this way, the amount of text that can be rendered by the data generator can be increased considerably. Moreover, to ensure that the rare characters are not underrepresented in the generated data, a counter for the number of occurrences of every individual character is maintained and used to guide the text extraction mechanism to choose text pieces containing the less frequently represented symbols.

The procedure described above is repeated until the number of occurrences of each character reaches a required minimum level $N_{\min}$, which guarantees that even rare characters are sufficiently well represented in each of the generated data sets, or until all available text from $\mathcal{T}$ has been processed. Note that this procedure uses sentences from a real text corpus (and not synthetically generated sequences of tokens) to ensure that the sampled character and n-gram distribution is the same as that of natural language texts.



al. Über diese Spornburg ist nur gt im 180°-Winkel zur "Stand"-P die ober

tegorie 3 der DB Station&Service det sich dort ein Radweg. Übe sper

ogen, bei der eine potenzielle Übertragung nschließend wird eine Prc

von der Gewebespende. Um eine Ü Zulassung nach § 21 bedürfen

t ein Denkmal an die Überführung Österreich. Heute ist sie ein

Bogenausbau im Über fusella" in Europa in Ö hr 1988 nahm die Band

Als Vorsitzender einer Ästhetikkommission in se der Öffentlichkeit für Denk

in New York. In dieser Zeit er hland sämtliche Ämter. Im Ruhrg

und Mündü [muh]. </doc> <doc id="658395 t Mayogo [mdm] u

lgarien und Wu Yunbilige, China nach Punkten u te er beim Dan Ko

de Masse plus 10 %, in die la [lin] und vor allem der in] und

bff], Manza [mzv], N , Bofi [bff], Manza [mzv], Ngbaka [ng , Bofi [bff],

und Manza [ngg] bildet das Bonjo die östli a [nga] und Manza [

r & Faber mit Bildern des Weltra aums von NASA, ESA und JAXA. Am 28.

Figure 2.6: A portion of a synthetically generated document. Excerpts extracted from a text corpus are rendered using a randomly selected font. The corresponding attributes of each rendered string are stored in the document metadata and can be used to extract and augment the examples.

2.4.3 Data Augmentation for OCR

To make the models more resistant to a common set of distortions, standard data augmentation methods are applied during training such as Gaussian smoothing, perspective distortions transformation, morphological filtering, upscaling, downscaling, additive noise induction, and elastic distortions transformation (Simard et al., 2003).

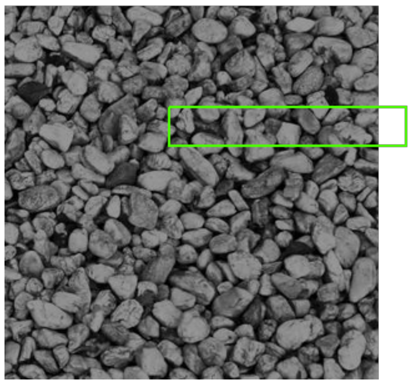
Additionally, to further improve robustness, a novel, task-specific data augmentation technique — *alpha compositing with background texture images* — is proposed. Specifically, each time a specific sample is presented to the network, it is composited (Porter and Duff, 1984) with a randomly selected background texture image as follows:

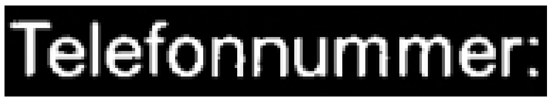
$$\mathcal{I}_{\text{res}}(x,y) = \mathcal{I}_{\text{src}}(x,y) + \alpha\frac{\mathcal{I}_{\text{bkgr}}(x,y)}{255}, \tag{2.4}$$

$$\alpha = 255 - \mathcal{I}_{\text{src}}(x,y), \tag{2.5}$$

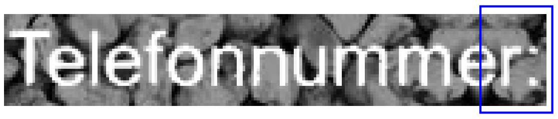
where x and y are pixel coordinates within an image and $\mathcal{I}_{\text{src}}$, $\mathcal{I}_{\text{bkgr}}$, and $\mathcal{I}_{\text{res}}$ are source, background, and result grayscale images, respectively. Note that, in Equations (2.4) and (2.5), we assume that each image is stored with 8 bits per sampled pixel, which accounts for 256 different intensities.

Note that the position of a background segment used for alpha compositing is selected randomly within the selected texture image. In the case that the selected background segment is not located completely within the texture image, e.g., because the dimensions of the source text line image exceed the dimensions of the texture image, the pixels of the texture image are mirrored about the image border with duplication (Bailey, 2011). Figure 2.7 illustrates the proposed data augmentation method.


(a) Background Texture



(b) Original Image



(c) Blended Image

Figure 2.7: Illustration of the proposed data augmentation method: **(a)** A background texture image. **(b)** A binarized text line image. **(c)** The result of blending the text line image with a background texture. The green box marks the region of the texture image that was randomly selected to be used for compositing. Note that it exceeds the dimensions of the image, thus the missing background content is mirrored with duplication, as can be seen in the resulting blended image (marked with a blue box).

By randomly altering the backgrounds of training samples, the network model is guided to focus on the features that facilitate text extraction and help to ignore background noise. Figure 2.8 presents several augmented examples that were generated by applying the proposed data augmentation pipeline on synthetically rendered text lines images.

In the presented system, data augmentation is dynamically applied to the input images both during training and inference. In contrast to the approach proposed by Jaderberg et al. (2014), the presented method renders undistorted synthetic documents once and then applies random data augmentations dynamically during inference. This enables efficient generation of samples and eliminates the significant overhead caused by input/output hard disk operations.

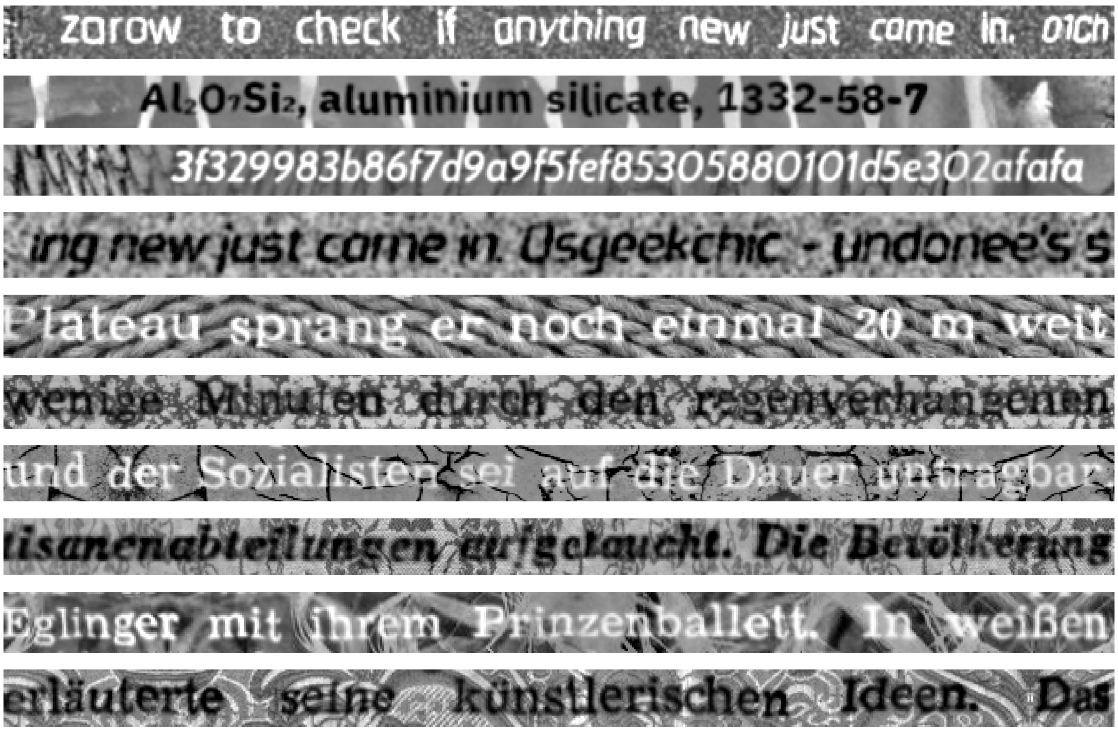

zorow to check if anything new just came in. 01Ch
Al₂O₇Si₂, aluminium silicate, 1332-58-7
3f329983b86f7d9a9f5fef85305880101d5e302afafa
ing new just came in. Osgeekchic - undonee's s
Plateau sprang er noch einmal 20 m weit
wenige Minuten durch den regenverhangenen
und der Sozialisten sei auf die Dauer untragbar.
tisanenabteilungen aufgetaucht. Die Bevölkerung
Eglinger mit ihrem Prinzenballett. In weißen
erläuterte seine künstlerischen Ideen. Das

Figure 2.8: Training and validation data samples from the proposed system that were generated using the data augmentation pipeline described in Section 2.4.3.

2.4.4 Geometric Normalization

Breuel (2017) recommended that text line images should be geometrically normalized before recognition. The normalization process used in this chapter is performed per text line before the feature extraction. To this end, the *baseline* and the *x-height* of the corresponding text line are used.

In typography, the baseline is an imaginary line upon which most letters sit, whereas the x-height is the mean line of lower-case letters (Evans et al., 2013, p. 29). During training, the text line attributes from the document metadata are used, whereas, at inference time, the layout analysis algorithm provides the baseline position and the x-height values for each line. Using the baseline information, the skew of the text lines is corrected, and the scale of each image is normalized, so that the x-height, as well as the heights of ascender and descender regions, are consistent across examples, as illustrated in Figure 2.9.

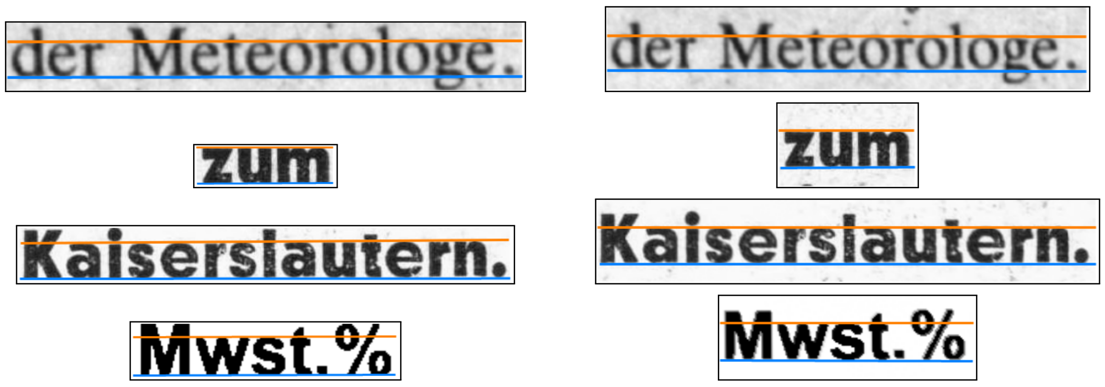


Figure 2.9: Examples of geometrically normalized text samples. Blue and orange lines correspond to the baselines and the x-height lines, respectively. **Left column**: Original examples extracted from a document image based on the bounding boxes of the text blocks. Note that each bounding box exactly fits the content of a text block, e.g., in the second row, the ascender and descender parts are missing and in the third and fourth row, there is no descender part. **Right column**: Geometrically normalized examples. The x-height, as well as the heights of ascenders and descenders, are consistent across different examples, which reduces variation in the vertical direction.

For the case where the normalization is not used, this procedure is skipped entirely, and each cropped text line sample is further passed on to the feature extractor. This case is especially relevant for scene text images, where normalization information is usually unavailable and expensive to compute separately.

2.5 Experimental Setup

2.5.1 Data Sources and Data Sets

To train and evaluate the proposed text recognition system, several data sets consisting of both real and synthetic documents were prepared. This section describes each of them in detail.

Real Documents Subset

For evaluation, 16 pages of scanned historical and recent German-language newspapers, as well as 11 contemporary German invoices, were collected. All documents were deskewed and preprocessed via the document layout analysis algorithms (Konya, 2013), providing us with the geometrical and logical document structure, including bounding boxes, baseline positions, and x-height values for each text line. The initial transcriptions obtained using the Tesseract OCR engine (Smith, 2007) were manually corrected. The ground-truth transcriptions were stored along with document layout information in Extensible Markup Language (XML) format.

Documents with Synthetically Generated Text

The core data set was extended by generating multiple further documents using the procedure described in Section 2.4.2. To this end, two large text corpora $\mathcal{T}$ were collected, namely, the English and German Wikipedia dump files[4], that were used as the sources of sentences for generating training data. For validation and test purposes, a corpus from the Leipzig Corpora Collection (Goldhahn et al., 2012) was used. Moreover, over 2,000 serif, sans serif, and monospace fonts[5] were collected as the set of fonts $\mathcal{F}$ required by the generation procedure. Furthermore, N_{min}, the minimum number of occurrences of each symbol in the data, was set to 5,000, 100, and 100 in the case of the synthetically generated training, validation, and test sets, respectively.

Summary of Raw Data Sources

The data sources used in the experiments are summarized in Table 2.1. The presented system was trained to recognize 132 different character classes, including basic lower- and uppercase Latin letters, whitespace characters, German umlauts, ligature ß, digits, punctuation marks, subscripts, superscripts, as well as mathematical, currency, and other commonly used symbols. The training part consists of about 6.4 million characters, 95.9% of which were synthetically generated.

Table 2.1: Statistics of the data sources used in the experiments.

Element	Newspapers		Invoices	Synthetic documents		
	Training	Test	Test	Training	Validation	Test
Documents	12	4	11	390	10	9
Text lines	7,049	1,943	486	200,030	4,827	4,650
Words	39,218	9,735	2,143	1,181,760	30,097	29,004
Text length	260,041	66,327	15,374	6,128,394	151,653	141,280

[4] https://dumps.wikimedia.org
[5] https://fonts.google.com

Evaluation Data Sets

To test the recognition accuracy of all examined OCR engines on the documents with standard quality, the evaluation using the documents from the corresponding test sets was performed. Moreover, the core test data set was extended by generating multiple distorted variants of each original document. To this end, synthetic distortions were induced to the input images prior to recognition.

Two different variants of the degradation were explored. In the first scenario (referred to as *moderate*), only geometrical transformations, morphological operations, blur, noise addition, upscaling, and downscaling were used. This scenario corresponds to the typical case of printed document scans of varying quality.

In the second scenario (referred to as *difficult*), all extracted text line images were additionally composited with a random background texture (Figure 2.8). Note that a different set of textures was used for training and during evaluation. Moreover, the image gray values were randomly inverted to test the ability of each method to handle examples with white text on a dark background and vice versa. The second scenario best corresponds to scene text recognition and was applied only in the case of synthetically generated documents.

In the case of the *moderate* and the *difficult* evaluation scenario, 30 randomly distorted copies of each original document were generated using different hyperparameters and varying the strength of each distortion. Note that since the distortions were applied randomly, some images obtained by this procedure may end up nearly illegible, even for human readers. Table 2.2 presents the ground truth length of each test subset. In summary, the employed evaluation data set consists of over eleven million characters.

Table 2.2: The length of the ground truth text from the evaluation data set.

Newspapers		Invoices		Synthetic documents		
Original	Moderate	Original	Moderate	Original	Moderate	Difficult
66,327	1,989,810	15,374	461,220	141,280	4,238,400	4,238,400

2.5.2 Training Setup

Generation of Training and Validation Samples

The batches containing the final training and validation samples were generated on the fly, as in the following. Text line images were randomly selected from the corresponding (training or validation) data set and the associated layout information was used to normalize each sample. Note that the normalization step is optional (see also Section 2.4.4), since especially in the case of scene text, it may be too computationally expensive and error-prone to extract exact baselines and x-heights at inference time.

Subsequently, all samples were rescaled to a fixed height of 32 pixels, while maintaining the original aspect ratio. This particular choice for the sample height was determined experimentally. Larger sample heights did not improve recognition accuracy for skew-free text lines. However, if the targeted use case involves the recognition of relatively long, free-form text lines, the use of larger sample heights should be considered.

Since text line lengths vary greatly, the corresponding images must be (zero) padded appropriately to fit the widest element within the batch. The amount of padding was minimized by composing batches from text lines having similar widths. Subsequently, random data augmentation methods were dynamically applied to each sample (Section 2.4.3). Finally, all pixel values were normalized to the range between 0.0 and 1.0 prior to feeding them into the deep learning model.

Implementation Details

To improve efficiency, an optimized training and validation data generation mechanism was implemented by employing the producer-consumer pattern. The producer process generates batches of samples and puts them into a shared queue. Concurrently, the consumer process removes generated batches from the queue and feeds them to the network. This mechanism allows for efficient utilization of available resources and simultaneously saves disc space used to store the data.

The deep learning-based models were developed in TensorFlow (Abadi et al., 2016). The Python interface of TensorFlow was used to implement the training process and the C++ interface was employed to perform the inference timings. As in the case of the sample generation, an efficient, multithreaded procedure was employed by using a lock-free, single-producer, single-consumer queue to handle the generation and recognition of samples in parallel.

Training Hyperparameters

All models were trained via mini-batch stochastic gradient descent using the *adaptive moment estimation* optimization method (Kingma and Ba, 2014). The learning rate was decayed by a factor of 0.99 every 1,000 iterations and the initial value of 0.0006 and 0.001 was used in the case of the hybrid and the FCN-based models, respectively. Moreover, *batch normalization* (Ioffe and Szegedy, 2015) was applied to speed up the training, and the dropout method (Srivastava et al., 2014) was employed to reduce the *overfitting* problem. Furthermore, the hybrid models were trained for approximately 300 epochs and the FCN-based models for about 500 epochs.

2.5.3 Evaluation Setup

The accuracy of the approaches examined in the experiments was compared using the character error rate (CER) metric, which was computed as the *Levenshtein distance* (Levenshtein, 1966) between the ground-truth and the recognized text line content. Note that the ground-truth text lines were extracted from the documents from the evaluation data sets (Section 2.5.1) using the available ground-truth layout information.

Baseline Approaches

Two established commercial OCR products were examined: ABBYY FineReader[6] v12 and OmniPage Capture SDK[7] v20.2, as well as a popular open-source OCR library — Tesseract (Smith, 2007) v3.04 and v4.0[8]. The latest Tesseract engine uses deep learning-based models similar to the proposed architecture.

The Method Proposed in This Work

The models with the architecture proposed in this chapter (Section 2.4.1), unless otherwise stated, are fine-tuned with real data (Table 2.1). Moreover, by default, they use geometric text line normalization (Section 2.4.4) and data augmentation methods (Section 2.4.3) except for elastic distortions.

External Language Models

Since it was shown that LSTMs learn an implicit language model (Sabir et al., 2017), the proposed system was evaluated without external language models or lexicons, although their use can likely further increase accuracy. By contrast, both

[6] `https://www.abbyy.com/en-eu/ocr-sdk`
[7] `https://www.nuance.com/print-capture-and-pdf-solutions/optical-character-recognition/omnipage/omnipage-for-developers.html` (Retrieved: 01/20/2019)
[8] `https://github.com/tesseract-ocr/tesseract/wiki/4.0-with-LSTM`

examined commercial OCR engines used language models and lexicons for English and German, and their settings have been chosen for best recognition accuracy. Moreover, the fast integer models[9] were used in the case of Tesseract 4.0 because they demonstrate a comparable running time to the other examined methods.

2.6 Experimental Results

2.6.1 Recognition Accuracy

Table 2.3 compares CERs of all examined OCR engines. The results show that the models with the proposed architecture outperform all other engines in terms of recognition accuracy in all evaluation scenarios. A substantial difference can be observed on distorted documents composited with background textures (see the results in the *difficult* category), where Tesseract and both commercial engines exhibit a very poor recognition accuracy. Noisy backgrounds hinder their ability to perform an adequate character segmentation. Although Tesseract 4.0 was trained on augmented synthetic data, we observe that it cannot properly deal with significantly distorted inputs.

A further qualitative examination of the results of the examined baseline engines revealed that these methods have difficulties in handling fonts with different, alternating styles located on the same page. Moreover, these engines have problems recognizing subscript and superscript symbols.

2.6.2 Analysis of the Most Frequent Errors

Table 2.4 gives an insight into the most frequent errors (insertions, deletions, and substitutions of characters) made by the best-performing methods on real versus synthetic data. All tested methods have the most difficulties in recognizing the exact number of whitespace characters due to the nonuniform letter and word spacing (e.g., kerning and justified text) across documents. This problem is particularly visible in the case of the real documents that were manually annotated, where a certain degree of ambiguity due to human judgment becomes apparent.

The remaining errors for the hybrid models include substitutions of similarly looking characters and are primarily focused on small or thin letters and symbols, which are indeed the ones most affected by distortions and background patterns. In contrast, ABBYY FineReader exhibits a clear tendency to insert spurious characters, especially for highly textured and distorted images.

[9] `https://github.com/tesseract-ocr/tessdata_fast`

Table 2.3: Character error rate (%) on the evaluation data sets. The results of the baseline OCR engines and multiple variants of the proposed architecture are reported. Different variants are denoted with superscripts: (1) *Hybrid* CNN-BLSTM-based models. (2) Models trained exclusively with synthetic data (*Synth*). (3) Models that employed elastic distortions during training (*Elastic*). (4) The FCN-based model (*FCN*). (5) The model trained without geometric normalization (*NoGeomNorm*) and without the alpha compositing augmentation method (*NoAlphaComp*). (6) The model that employed either peephole LSTM units (*Peephole*) or GRU RNN cells (*GRU*). **Bold** values indicate the best results within each evaluation data set (Section 2.5.1).

Method	Newspapers		Invoices		Synthetic documents		
	Original	Moderate	Original	Moderate	Original	Moderate	Difficult
ABBYY FineReader	0.63	3.03	2.65	4.38	6.64	10.86	19.27
OmniPage Capture	0.31	3.76	2.61	9.94	7.62	17.13	58.43
Tesseract 3.04	1.15	16.90	6.11	10.16	11.79	17.76	37.00
Tesseract 4.0	1.14	9.63	4.53	6.66	8.70	14.80	35.91
Proposed$^{\text{FCN}}$	0.16	0.81	1.66	3.28	1.03	2.02	3.25
Proposed$^{\text{Hybrid}}$	**0.11**	0.75	**1.63**	**2.33**	0.62	1.48	2.84
Proposed$^{\text{Hybrid, NoGeomNorm}}$	0.13	1.14	2.86	3.67	**0.46**	**1.31**	4.50
Proposed$^{\text{Hybrid, Peephole}}$	0.14	0.69	1.85	2.81	0.53	1.35	**2.43**
Proposed$^{\text{Hybrid, GRU}}$	0.16	**0.67**	2.14	3.31	1.10	2.02	4.16
Proposed$^{\text{Hybrid, Elastic}}$	**0.11**	0.73	**1.63**	2.47	0.65	1.46	2.70
Proposed$^{\text{Hybrid, NoAlphaComp}}$	0.14	1.24	2.49	3.58	0.48	1.94	7.54
Proposed$^{\text{Hybrid, Synth}}$	0.20	0.96	1.85	3.05	0.67	1.58	2.84
Proposed$^{\text{Hybrid, Synth, Elastic}}$	0.20	0.95	1.76	2.89	0.69	1.51	2.69

Table 2.4: Top 10 most frequent errors for the best OCR engines in the main experiment (Table 2.3). Note that the real data (left side) comprises both newspapers and invoices (Table 2.1).

(a) Proposed$^{\text{Hybrid}}$ (real data)

Error type	Rate [%]
Insertion of ' '	25.77
Substitution 'l'→'i'	4.43
Substitution '.'→','	3.56
Insertion of '.'	2.77
Substitution 'i'→'l'	2.67
Insertion of '_'	2.52
Substitution 'I'→'l'	2.15
Insertion of 't'	1.27
Substitution 'o'→'a'	1.21
Substitution 'f'→'t'	1.19

(b) Proposed$^{\text{Hybrid, Peephole}}$ (synthetic data)

Error type	Rate [%]
Insertion of ' '	8.53
Substitution '0'→'O'	4.09
Deletion of '.'	1.77
Substitution 'O'→'Ö'	1.62
Deletion of '_'	1.58
Deletion of '-'	1.45
Substitution 'I'→'l'	1.44
Substitution '.'→','	1.33
Insertion of 'r'	1.00
Substitution '©'→'O'	0.97

(c) ABBYY FineReader (real data)

Error type	Rate [%]
Insertion of ' '	12.35
Insertion of '.'	5.70
Substitution ','→'.'	2.95
Insertion of 'i'	2.70
Insertion of 'r'	2.09
Deletion of 'e'	1.83
Substitution 'c'→'e'	1.77
Insertion of 'l'	1.63
Insertion of 'n'	1.41
Insertion of 't'	1.35

(d) ABBYY FineReader (synthetic data)

Error type	Rate [%]
Insertion of ' '	9.27
Insertion of 'i'	1.72
Insertion of 'e'	1.55
Insertion of 't'	1.33
Insertion of 'r'	1.21
Insertion of '.'	1.19
Insertion of 'l'	1.18
Insertion of '-'	1.07
Insertion of 'n'	1.06
Insertion of 'a'	0.96

2.6.3 Runtime Analysis

Figure 2.10 presents the runtime comparison. All baseline approaches and selected models with the proposed architecture are included. As can be seen in the results, both commercial engines and Tesseract 3.04 work slowly for significantly distorted images. Apparently, they make use of certain computationally expensive image restoration techniques to be able to handle low-quality inputs.

Unsurprisingly, the models with the proposed architecture that were executed on a graphics processing unit (GPU) are the fastest across the board. All experiments were conducted on a workstation equipped with an Nvidia GeForce GTX 745 graphics card and an Intel Core i7-6700 central processing unit (CPU).

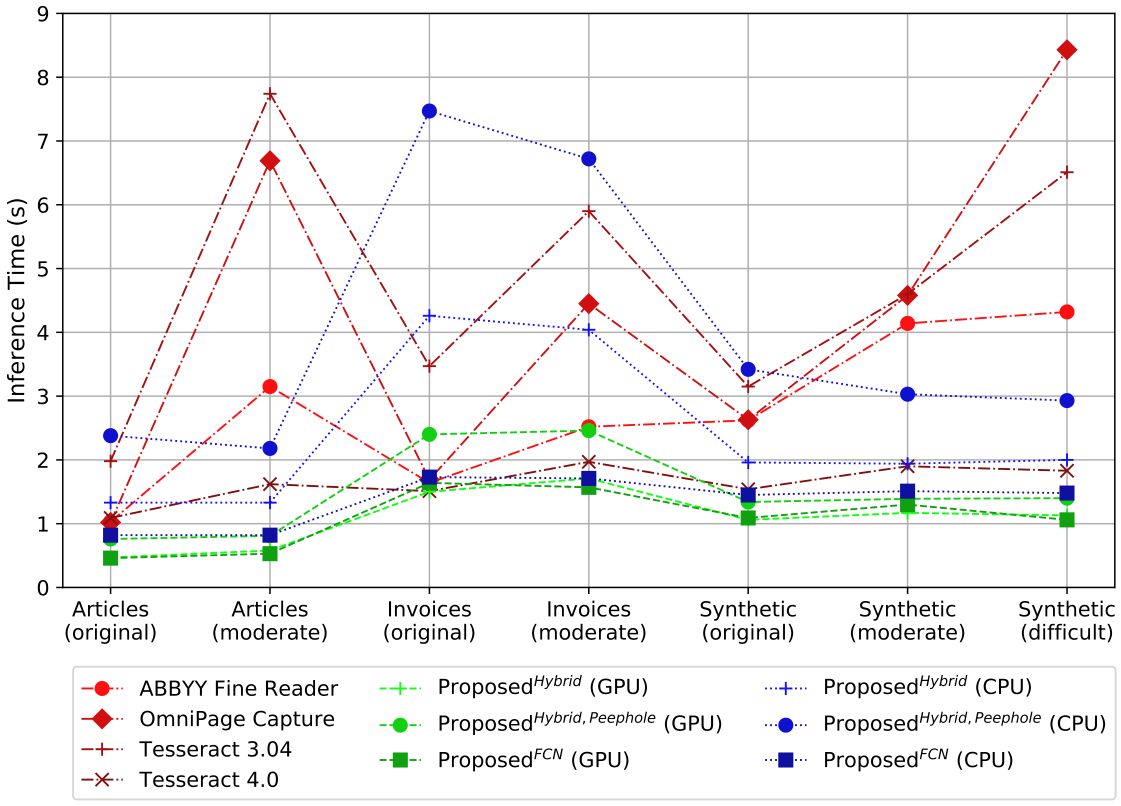


Figure 2.10: Runtime comparison (in seconds) on the test data sets per standard page with 1,500 symbols. Values for the documents from the *original* category as well as the CPU experiments are averaged over 10 trials. All other values are averaged over 30 trials. Moreover, the CPU experiments use a batch size of four images, whereas the GPU runs use a batch of 48 images. Note that data points from different data sets are connected solely to allow easier traceability of the results of each engine. Best viewed in color.

2.6.4 Qualitative Example of CTC Decoding

Figure 2.11 shows an example of the CTC decoding procedure on a real-world example. The visualization gives an insight into the decoding process and helps greatly in spotting many decoding failures.

For instance, the text line presented in this example was recognized by the proposed system with one minor error as "`tagtäglich viele gute ldeen und`". When we closely look at probabilities in a region where this error occurs, we notice that probabilities of the letters '`l`' and '`I`' are very close, as both characters are also visually similar, though. Such ambiguities can be potentially detected and corrected for example via an external language model.

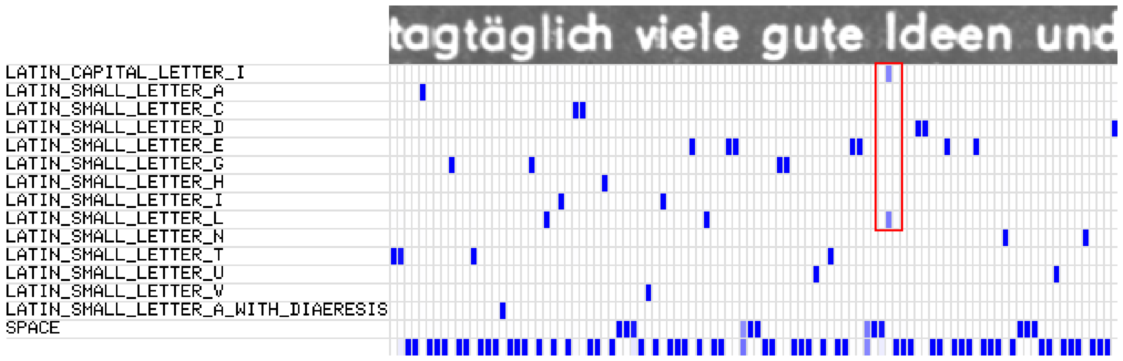


Figure 2.11: Visualization of the CTC decoding results on an example from the data collection used in the experimental evaluation. Each cell in the grid represents the probability of a specific character at a specific time step — the darker the color, the higher the probability. The last row represents the probability of a blank character that separates two consecutive output symbols. The text line presented in this example was erroneously recognized as "`tagtäglich viele gute ldeen und`". The probabilities of the letters '`l`' and '`I`' are very close in this case (see a red box in the image), as both characters are visually similar, which may cause incorrect predictions in such cases.

2.6.5 Ablation Study

In this section, the impact of different model components on the recognition performance of the proposed system is analyzed.

FCN-Based vs. Hybrid CNN-BLSTM Models

The fully convolutional model (Proposed$^{\mathrm{FCN}}$ in Table 2.3) achieves a slightly lower accuracy than the best hybrid variant. However, its inference time is significantly lower on the CPU. This clearly shows that convolutional layers are much more amenable to parallelization than recurrent units.

GRU vs. LSTM Cells

The model employing GRU units (Proposed$^{\text{Hybrid, GRU}}$ in Table 2.3) exhibits competitive but slightly lower accuracy than the LSTM variants. Since the GRU cell has a simpler structure than the basic LSTM cell, it should work a bit faster on the CPU, which is consistent with the obtained results.

Peephole Connections

The model utilizing peephole LSTM cells (Proposed$^{\text{Hybrid, Peephole}}$ in Table 2.3), which also pools feature maps along the width dimension only once, i.e., omits the second max pooling operation in Table A.1, exhibits better recognition accuracy in the scene text scenario. This is not the case for the original and moderately distorted document scans, where the peepholes do not seem to bring any additional accuracy gains compared to the vanilla LSTM model. The use of peephole connections does, however, add a significant runtime overhead.

Alpha Compositing with Background Textures Data Augmentation

The model that was trained without using the proposed data augmentation technique (Proposed$^{\text{Hybrid, NoAlphaComp}}$ in Table 2.3), i.e., alpha compositing with background textures (Section 2.4.3), exhibits significantly higher error rates, not only on samples with complicated backgrounds but also on those with significant distortions. This confirms that the proposed augmentation technique has a positive effect on the robustness of neural OCR models.

Geometric Normalization

The model that did not use geometric normalization (Proposed$^{\text{Hybrid, NoGeomNorm}}$ in Table 2.3), as described in Section 2.4.4, exhibited a decrease in accuracy, especially in the case of images with stronger distortions (see the results in the *difficult* category in Table 2.3). This indicates that geometric normalization is indeed beneficial, but not indispensable. Apparently, max pooling and strided convolution operations provide enough translational invariance in most cases.

Training With Synthetic Data Only

The models that were trained exclusively using synthetic data (Proposed$^{\text{Hybrid, Synth}}$ and Proposed$^{\text{Hybrid, Synth, Elastic}}$ in Table 2.3) obtained very competitive results, which indicates that using such a model together with proper data augmentation is sufficient for achieving a satisfactory recognition accuracy.

Elastic Distortions Data Augmentation

Apparently, nonlinear distortions can further reduce the error rate of models, particularly those trained exclusively on synthetic data (cf. Proposed$^{\text{Hybrid, Synth}}$ versus Proposed$^{\text{Hybrid, Synth, Elastic}}$ in Table 2.3). Hence, this augmentation method is beneficial, especially in cases where annotated real data is not available or simply too difficult to produce. We also observe that although most of the models were trained without elastic distortions applied to training data, they can nonetheless deal with test data augmented with nonlinear distortions. We can attribute this to the fact that the synthetic training data was generated using a substantial number of fonts, which allowed for reasonable variation of text styles.

2.7 Summary

In this chapter, a general text line recognition method was proposed (Section 2.4). Experiments under different scenarios, on both real and synthetically generated documents, showed that both developed architectures — the hybrid CNN-BLSTM and the FCN-based model — outperform leading commercial and open-source engines (Section 2.6). In particular, the presented approach demonstrated an outstanding recognition accuracy on severely degraded inputs.

The proposed architecture is universal and can be employed to recognize printed, handwritten, or scene text (Section 2.4.1). The training of models for other languages is straightforward. Moreover, external language models or lexicons can be easily integrated into the CTC decoding process. Via the proposed pipeline, deep neural network models can be trained using only text line-level annotations. This saves a considerable manual annotation effort, previously required for producing the character- or word-level ground-truth segmentation and the corresponding transcription (Section 2.3.1).

A novel data augmentation technique — alpha compositing with background textures — was introduced in Section 2.4.3 and evaluated with respect to its effects on the overall recognition robustness. Moreover, the analysis in Section 2.6.5

confirmed that generating synthetic training data is indeed a viable and scalable alternative to collecting real training data, provided that sufficiently diverse samples can be generated by the data augmentation modules.

The effect of different structural choices and data augmentation on recognition accuracy and inference time was experimentally investigated (see Section 2.6.1 and Section 2.6.5). Hybrid recognition architectures proved to be more accurate, however, they are considerably more expensive in terms of computational complexity than the purely convolutional approaches. The peephole LSTM units exhibited better accuracy in more challenging scenarios but also induced significant runtime overhead. On the other hand, the GRU cells provide competitive accuracy and work efficiently on the CPU, which makes them a viable solution in the scenarios, where execution speed is critical. Nevertheless, the basic LSTM units offer the best trade-off between accuracy and runtime and proved to be the most universal in all text line recognition scenarios.

Future Work Directions

The importance of a solid data generation pipeline cannot be overstated. As such, future work on improving the proposed data augmentation method should involve its continuous improvement and comparison with other notable efforts from the research community, such as the work of Jaderberg et al. (2014).

Moreover, the FCN approaches, in particular, offer great potential for future improvement. The incorporation of advances in the field of computer vision, such as squeeze-and-excitation blocks (Hu et al., 2018) should provide further improvements in terms of both recognition accuracy and robustness. Another possible extension would be to integrate an automatic rectification module as in Jaderberg et al. (2015) to handle irregular text.

3 Noise-Aware Training

Preface

This chapter is adapted from Namysl, Behnke, and Köhler (2020)[1], previously published by the Association for Computational Linguistics and presented at the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)[2]. Moreover, Section 3.6 incorporates related work presented in Namysl, Behnke, and Köhler (2021)[3], previously published by the Association for Computational Linguistics and presented at the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021)[4].

The content from both aforementioned publications is used under the terms of Creative Commons Attribution 4.0 International License.[5]

Statement of Personal Contribution

The author of this thesis substantially contributed to all aspects of the previous publications (Namysl, Behnke, and Köhler, 2020, 2021), including the conception, design, and implementation of the proposed methods, the preparation of the data for training and evaluation of the proposed approach, conducting the experimental evaluation, the analysis and interpretation of the experimental results, drafting the manuscript, as well as the revision and final approval of the version to be published.

The content presented in this chapter, unless otherwise stated, is the contribution of the author of this thesis.

[1] ©2020 Association for Computational Linguistics. Reprinted in Appendix C.2.
[2] `https://acl2020.org`
[3] ©2021 Association for Computational Linguistics. Reprinted in Appendix C.3.
[4] `https://2021.aclweb.org`
[5] `https://creativecommons.org/licenses/by/4.0`

Unpublished Content

Compared to the previous publication, additional, unpublished content is included in this chapter. In particular, in addition to the evaluation of the named entity recognition task, the evaluation results in the part-of-speech tagging and syntactic chunking scenarios are additionally presented in Section 3.5.2.

Abstract

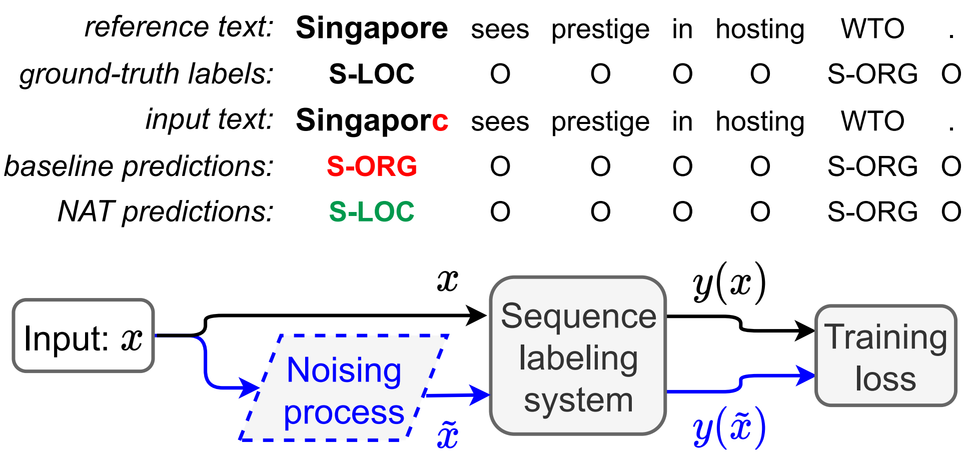
Sequence labeling systems should perform reliably not only under ideal conditions but also with corrupted inputs, as these systems often process user-generated text or follow an error-prone upstream component. In this chapter, the noisy sequence labeling problem is formulated, which assumes that the input may undergo an unknown noising process. Two noise-aware training objectives are proposed to improve the robustness of sequence labeling performed on perturbed input: The *data augmentation* method trains a neural model using a mixture of clean and noisy samples, whereas the *stability training* algorithm encourages the model to create a noise-invariant latent representation. Extensive experiments on the original English and German sequence labeling benchmarks as well as on their variants perturbed with real OCR errors and misspellings confirm that the proposed noise-aware training method consistently improves the robustness of the state-of-the-art sequence labeling models, preserving accuracy on the original, error-free input.

3.1 Introduction

Sequence labeling is a task that involves the assignment of a categorical label to the elements of an input sequence of natural language tokens. Many components of the IE pipeline (Figure 1.1) can be formulated as a sequence labeling problem, e.g, NER classifies the input tokens into some predefined entity classes.

Sequence labeling systems are generally trained using error-free input text, although, in real-world scenarios, they often follow an error-prone upstream component, such as OCR (Neudecker, 2016) or ASR (Parada et al., 2011) engine. Sequence labeling is also often performed on user-generated text, which may contain spelling mistakes or typos (Derczynski et al., 2013). Errors introduced in an upstream task are propagated downstream, diminishing the performance of the end-to-end IE system (Alex and Burns, 2014).

While humans can easily cope with typos, misspellings, and the complete omission of letters when reading (Rawlinson, 2007), most NLP systems fail when processing corrupted or noisy text (Belinkov and Bisk, 2018). Although this problem is not new to NLP, only a few works addressed it explicitly (Karpukhin et al., 2019; Piktus et al., 2019). Other methods that do not account for the errors in the input must rely on the noise that occurs naturally in the training data.



reference text:	**Singapore**	sees	prestige	in	hosting	WTO	.
ground-truth labels:	**S-LOC**	O	O	O	O	S-ORG	O
input text:	**Singaporc**	sees	prestige	in	hosting	WTO	.
baseline predictions:	**S-ORG**	O	O	O	O	S-ORG	O
NAT predictions:	**S-LOC**	O	O	O	O	S-ORG	O

Figure 3.1: An example of a labeling error on a slightly perturbed sentence. The noise-aware method correctly predicted the location (LOC) label for the first word, as opposed to the standard approach, which misclassified it as an organization (ORG). The example is complemented with a high-level idea of the NAT method, where the original sentence and its noisy variant are passed together through the system. The final loss is computed based on both sets of features, which improves the robustness against the input perturbations.

In this chapter, we are primarily concerned with the performance difference of sequence labeling performed on error-free and noisy input. We aim to answer the following question: *Is it possible to narrow the gap between these two domains and design an approach that is transferable to different noise distributions at test time?* Figure 3.1 illustrates the problem and the approach developed toward improving the robustness of sequence labeling models.

Firstly, in Section 3.2.2, the noisy sequence labeling problem, which assumes that the input may undergo an unknown noising process, is formulated. Moreover, in Section 3.3.1, a model that can be used to estimate the real error distribution is introduced. Furthermore, in Section 3.3.2, a novel noise induction procedure used to simulate real-world noisy input is presented.

Secondly, inspired by recent research in computer vision (S. Zheng et al., 2016), NMT (Cheng et al., 2018), and ASR (Sperber et al., 2017), two noise-aware training (NAT) objectives are proposed. These objectives improve the accuracy of sequence labeling performed on noisy input without reducing the efficiency on the original, error-free data. To this end, a *data augmentation* algorithm was developed (Section 3.3.3). It directly induces noise into the input data to perform training of the neural model using a mixture of noisy and error-free samples. Moreover, a *stability training* method (S. Zheng et al., 2016) was implemented and adapted to the sequence labeling scenario. This method explicitly addresses the noisy data problem by encouraging the model to produce a noise-invariant latent representation of the input (Section 3.3.4).

Finally, a thorough evaluation of the proposed method against the state-of-the-art baseline models (Akbik et al., 2018; Devlin et al., 2019; Peters et al., 2018) was performed. The experiments on the text that contains real OCR errors and misspellings demonstrated the effectiveness of the presented approach (Section 3.5).

3.2 Background and Motivation

In this section, the neural sequence labeling task is described (Section 3.2.1), as well as its extension that includes the noise present on the source side of the sequence labeling system (Section 3.2.2). Moreover, in Section 3.2.3, the research problem studied in this chapter is formulated.

3.2.1 Neural Sequence Labeling

Figure 3.2 presents a typical architecture for the neural sequence labeling task. We will refer to the sequence labeling system as $F(x;\theta)$, abbreviated as $F(x)$[6], where $x = (x_1, \ldots, x_N)$ is a tokenized input sentence of length N, and θ represents all learnable parameters of the system. $F(x)$ takes x as input and outputs the probability distribution over the class labels $y(x)$ as well as the final sequence of labels $\hat{y} = (\hat{y}_1, \ldots, \hat{y}_N)$.

Either a softmax model (Chiu and Nichols, 2016) or a conditional random field (CRF) (Lample et al., 2016) can be used to model the output distribution over the class labels $y(x)$ from the logits $l(x)$, i.e., non-normalized predictions, and

[6] We drop the θ parameter for brevity in the remainder of the chapter. Nonetheless, we still assume that all components of $F(x;\theta)$ and all expressions derived from it also depend on θ.

to output the final sequence of labels $\hat{y}$. As a labeled entity can span several consecutive tokens within a sentence, special tagging schemes are often employed for decoding, e.g., BIOES, where the **B**eginning-, **I**nside-, **O**utside-, **E**nd-of-entity, and **S**ingle-tag-entity subtags are also distinguished (Ratinov and Roth, 2009). This method introduces strong dependencies between subsequent labels, which are modeled explicitly by a CRF (Lafferty et al., 2001) that produces the most likely sequence of labels.

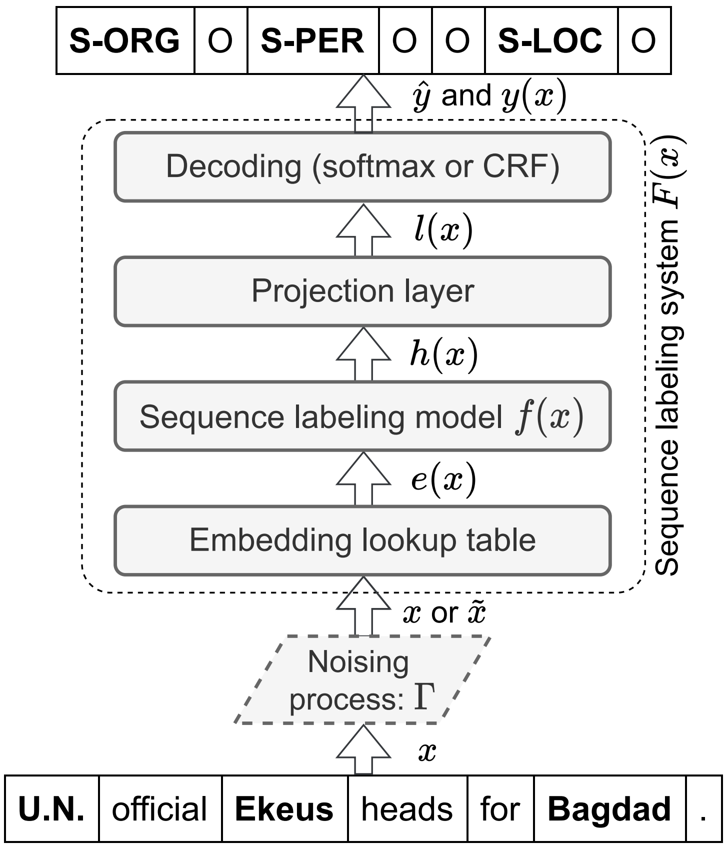


Figure 3.2: Neural sequence labeling architecture. In the standard scenario (Section 3.2.1), the original sentence x is fed as input to the sequence labeling system $F(x)$. Token embeddings $e(x)$ are retrieved from the corresponding lookup table and fed to the sequence labeling model $f(x)$, which outputs latent feature vectors $h(x)$. The latent vectors are then projected to the class logits $l(x)$, which are used as input to the decoding model (softmax or CRF) that outputs the distribution over the class labels $y(x)$ and the final sequence of labels $\hat{y}$. In a real-world scenario (Section 3.2.2), the input sentence undergoes an unknown noising process Γ, and the perturbed sentence $\tilde{x}$ is fed to the sequence labeling system $F(x)$.

3.2.2 Noisy Neural Sequence Labeling

Similar to human readers, sequence labeling should perform reliably both in ideal and suboptimal conditions. Unfortunately, this is rarely the case. User-generated text is a rich source of informal language containing misspellings, typos, or scrambled words (Derczynski et al., 2013). Noise can also be introduced in an upstream task, like OCR (Alex and Burns, 2014) or ASR (Chen et al., 2017), causing the errors to be propagated downstream.

The sequence labeling model introduced in Section 3.2.1 is defined as a transformation of the source sequence of tokens x to the target sequence of labels y. To include the noise present on the source side of $F(x)$, we can modify its definition accordingly (Figure 3.2). Let us assume that the input sentence x is additionally subjected to some unknown noising process $\Gamma = P(\tilde{x}_i|x_i)$, where x_i is the original i-th token and $\tilde{x}_i$ is its distorted equivalent. Let $\mathcal{V}$ be the vocabulary of tokens and $\tilde{\mathcal{V}}$ be a set of all finite character sequences over an alphabet Σ. Γ is known as the *noisy channel matrix* (Brill and Moore, 2000) and can be constructed by estimating the probability $P(\tilde{x}_i|x_i)$ of each distorted token $\tilde{x}_i$ given the intended token x_i for every $x_i \in \mathcal{V}$ and $\tilde{x}_i \in \tilde{\mathcal{V}}$.

3.2.3 Problem Definition

In this chapter, we primarily study the effectiveness of state-of-the-art NER systems in handling imperfect input data. NER can be considered as a special case of the sequence labeling problem, where the goal is to locate all named entities mentioned in unstructured text and to classify them into predefined categories, e.g., person names, organizations, and locations (Tjong Kim Sang and De Meulder, 2003). Note that, although the evaluation of the proposed approach primarily focuses on the NER task, other sequence labeling scenarios are also experimentally validated (Sections 3.4.1 and 3.5.2).

Sequence labeling systems are often trained using error-free textual input. Consequently, they exhibit degraded performance in real-world scenarios where the transcriptions are produced by the previous upstream component, such as OCR or ASR modules (Section 3.2.2), which results in a detrimental mismatch between the training and the test conditions. The goal of the method proposed in this chapter is thus to improve the robustness of sequence labeling performed on data from noisy sources, without deteriorating performance on the original data.

Moreover, let us assume that the source sequence of tokens x may contain errors. However, the noising process is generally label-preserving, i.e., the level of noise is not significant enough to affect the corresponding labels[7]. It follows that the noisy token $\tilde{x}_i$ inherits the ground-truth label y_i from the underlying original token x_i.

3.3 Proposed Noise-Aware Training Method

In the previous section, we highlighted that the standard sequence labeling systems are not robust to perturbed input (Section 3.2.2). In this section, the noise model employed by the proposed method is introduced (Section 3.3.1). Moreover, the noise induction procedure (Section 3.3.2) and the proposed NAT objectives (Section 3.3.3 and Section 3.3.4) are thoroughly described.

3.3.1 Noise Model

To model the input noise, the character-level noisy channel matrix Γ is used, which we will refer to as the *character confusion matrix* (Section 3.2.2).

Natural Noise

The natural error distribution can be estimated by calculating the alignments between the pairs $(\tilde{x}, x) \in \mathcal{P}$ of noisy and clean sentences using the *Levenshtein distance* metric (Levenshtein, 1966), where $\mathcal{P}$ is a corpus of paired noisy and manually corrected sentences (Section 3.2.2). The allowed edit operations include insertions, deletions, and substitutions of characters. Insertions and deletions can be modeled by introducing an additional symbol ε into the character confusion matrix. The probability of insertion and deletion can then be formulated as $P_{ins}(c|\varepsilon)$ and $P_{del}(\varepsilon|c)$, where c is a character to be inserted or deleted, respectively.

Synthetic Noise

$\mathcal{P}$ is usually laborious to obtain. Moreover, the exact modeling of noise might be impractical, and it is often difficult to accurately estimate the exact noise distribution to be encountered at test time. Such distributions may depend on, e.g., the OCR engine used to digitize the documents. Therefore, the estimated natural error distribution is intended for evaluation, and a simplified, synthetic

[7] Moreover, a human reader should be able to infer the correct label y_i from the token $\tilde{x}_i$ and its context. We assume that this corresponds to a character error rate of $\leq 20\%$.

error model is used for training. In this model, all types of edit operations, i.e., character insertions P_{ins}, deletions P_{del}, and substitutions P_{subst}, are equally likely:

$$\sum_{\tilde{c}\in\Sigma\backslash\{\varepsilon\}} P_{ins}(\tilde{c}|\varepsilon) = P_{del}(\varepsilon|c) = \sum_{\tilde{c}\in\Sigma\backslash\{c,\varepsilon\}} P_{subst}(\tilde{c}|c), \tag{3.1}$$

where c and $\tilde{c}$ are the original and the perturbed characters, respectively. Moreover, P_{ins} and P_{subst} are uniform over the set of allowed insertion and substitution candidates, respectively.

More specifically, let η be a hyperparameter that controls the amount of noise to be induced with this method and $P_{edit} = \eta/3$ be the probability of performing a single character edit operation (insertion, deletion, or substitution) that replaces the source character c with a noisy character $\tilde{c}$, where $\tilde{c} \neq c$. Equation (3.2) defines the *vanilla* error distribution, which is used at training time:

$$P(\tilde{c}|c) = \begin{cases} \dfrac{P_{edit}}{|\Sigma\backslash\{\varepsilon\}|}, & \text{if } c=\varepsilon \text{ and } \tilde{c}\neq\varepsilon. & (3.2a)\\ 1-P_{edit}, & \text{if } c=\varepsilon \text{ and } \tilde{c}=\varepsilon. & (3.2b)\\ \dfrac{P_{edit}}{|\Sigma\backslash\{c,\varepsilon\}|}, & \text{if } c\neq\varepsilon \text{ and } \tilde{c}\neq c. & (3.2c)\\ P_{edit}, & \text{if } c\neq\varepsilon \text{ and } \tilde{c}=\varepsilon. & (3.2d)\\ 1-2P_{edit}, & \text{if } c\neq\varepsilon \text{ and } \tilde{c}=c. & (3.2e) \end{cases}$$

It consists of the following components:

(a) The *insertion probability* $P_{ins}(\tilde{c}|\varepsilon)$ in Equation (3.2a). It describes how likely it is to insert a nonempty character $\tilde{c}\neq\varepsilon$ and it is uniform over the set of all characters from the alphabet Σ, except the ε symbol.

(b) The *keep ε probability* $P_{keep}(\varepsilon|\varepsilon)$ in Equation (3.2b).

(c) The *substitution probability* $P_{subst}(\tilde{c}|c)$ in Equation (3.2c). It is uniform over the set of all characters from the alphabet Σ, except the source character c and the ε symbol.

(d) The *deletion probability* $P_{del}(\varepsilon|c)$ in Equation (3.2d).

(e) The *keep probability* $P_{keep}(c|c)$ in Equation (3.2e).

Equations (3.2a) and (3.2b) correspond to the row in the character confusion matrix Γ, where $c=\varepsilon$, and form a valid probability distribution:

$$P_{keep}(\varepsilon|\varepsilon) + \sum_{\tilde{c}\in\Sigma\backslash\{\varepsilon\}} P_{ins}(\tilde{c}|c) = 1. \tag{3.3}$$

Similarly, Equations (3.2c) to (3.2e) correspond to the rows in the character confusion matrix Γ, where $c \in \Sigma \backslash \{\varepsilon\}$, and are also valid probability distributions:

$$P_{del}(\varepsilon|c) + P_{keep}(c|c) + \sum_{\tilde{c}\,\in\,\Sigma\backslash\{c,\,\varepsilon\}} P_{subst}(\tilde{c}|c) = 1 \tag{3.4}$$

Finally, Figures A.1 and A.2 in the appendix depict the confusion matrices used in the proposed vanilla and OCR error models, respectively. Note that the former assigns equal probability to all edit operations and the latter is biased toward substitutions of characters with similar shapes.

3.3.2 Noise Induction Procedure

Ideally, we would use the noisy sentences annotated with named entity labels for training robust sequence labeling models. Unfortunately, such data is scarce. On the other hand, labeled, error-free text corpora are widely available (Benikova et al., 2014; Tjong Kim Sang and De Meulder, 2003). Hence, the method proposed in this chapter uses the standard NER corpora and induces noise into the input tokens during training synthetically.

In contrast to the image domain, which is continuous, the text domain is discrete, and we cannot directly apply continuous perturbations to written language. Although some works applied distortions at the level of embeddings (Bekoulis et al., 2018; Miyato et al., 2017; Yasunaga et al., 2018), we do not have a good intuition about how it changes the meaning of the underlying textual input. Instead, the proposed method employs the noise induction procedure to generate distorted copies of the input.

For every input sentence x, each token $x_i = (c_1, \dots, c_K)$ is independently perturbed, where K is the length of x_i, with the following procedure illustrated in Figure 3.3:

(1) The ε symbol is inserted before the first and after every character of x_i to get an extended token $x'_i = (\varepsilon, c_1, \varepsilon, \dots, \varepsilon, c_K, \varepsilon)$.

(2) For every character c'_k of x'_i, the replacement character $\tilde{c}'_k$ is sampled from the corresponding probability distribution $P(\tilde{c}'_k|c'_k)$, which can be obtained by taking a row of the character confusion matrix that corresponds to c'_k. As a result, we get a noisy version of the extended input token $\tilde{x}'_i$.

(3) All ε symbols are removed from $\tilde{x}'_i$ and the remaining characters are collapsed to obtain a noisy token $\tilde{x}_i$.

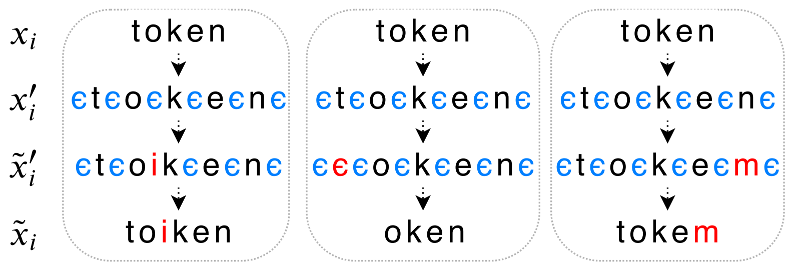


Figure 3.3: Illustration of the proposed noise induction procedure. Three examples correspond to insertion, deletion, and substitution errors. x_i, x'_i, $\tilde{x}'_i$, and $\tilde{x}_i$ are the original, extended, extended noisy, and noisy tokens, respectively.

3.3.3 Data Augmentation Objective

We can improve robustness to noise at test time by introducing various forms of artificial noise during training. We can distinguish between regularization methods such as dropout (Srivastava et al., 2014) and task-specific data augmentation that transforms the data to resemble noisy input. The latter technique was successfully applied in other domains, including computer vision (Krizhevsky et al., 2012) and speech recognition (Sperber et al., 2017).

During training, the proposed method artificially induces noise into the original sentences using the algorithm described in Section 3.3.2 and trains the neural sequence labeling models using a mixture of error-free and noisy sentences. Let $\mathcal{L}_0(x, y; \theta)$ be the standard training objective for the sequence labeling task, where x is the input sentence, y is the corresponding ground-truth sequence of labels, and θ represents the parameters of $F(x)$. The composite loss function that describes the data augmentation objective is defined as follows:

$$\mathcal{L}_{augm}(x, \tilde{x}, y; \theta) = \mathcal{L}_0(x, y; \theta) + \alpha \mathcal{L}_0(\tilde{x}, y; \theta), \tag{3.5}$$

where $\tilde{x}$ is the perturbed sentence, and α is the weight of the noisy loss component.

$\mathcal{L}_{augm}$ is a weighted sum of standard losses calculated using error-free and noisy sentences. Intuitively, the model that would optimize $\mathcal{L}_{augm}$ should be more robust to imperfect input data, retaining the ability to perform well on error-free text. Figure 3.4a presents a schematic visualization of the data augmentation approach.

3.3.4 Stability Training Objective

S. Zheng et al. (2016) pointed out the output instability issues of deep neural networks. They proposed a training method to stabilize deep networks against small input perturbations and applied it to the tasks of near-duplicate image detection, similar image ranking, and image classification. Inspired by their idea, in the proposed approach, the stability training method is adapted to the natural language scenario.

Our goal is to stabilize the outputs $y(x)$ of a sequence labeling system against small input perturbations, which can be thought of as flattening $y(x)$ in a close neighborhood of any input sentence x. When a perturbed copy $\tilde{x}$ is close to x, then $y(\tilde{x})$ should also be close to $y(x)$. Given the standard training objective $\mathcal{L}_0(x, y; \theta)$, the original input sentence x, its perturbed copy $\tilde{x}$, and the sequence of ground-truth labels y, the stability training objective $\mathcal{L}_{stabil}$ can be defined as follows:

$$\mathcal{L}_{stabil}(x, \tilde{x}, y; \theta) = \mathcal{L}_0(x, y; \theta) + \alpha \mathcal{L}_{sim}(x, \tilde{x}; \theta), \tag{3.6}$$

$$\mathcal{L}_{sim}(x, \tilde{x}; \theta) = \mathcal{D}\big(y(x), y(\tilde{x})\big), \tag{3.7}$$

where $\mathcal{L}_{sim}$ encourages the similarity of the model outputs for both x and $\tilde{x}$, $\mathcal{D}$ is a task-specific feature distance measure, and α balances the strength of the similarity objective.

Let $R(x)$ and $Q(\tilde{x})$ be the discrete probability distributions obtained by calculating the softmax function over the logits $l(x)$ for x and $\tilde{x}$, respectively:

$$R(x) = P(y|x) = softmax\big(l(x)\big), \tag{3.8}$$

$$Q(\tilde{x}) = P(y|\tilde{x}) = softmax\big(l(\tilde{x})\big). \tag{3.9}$$

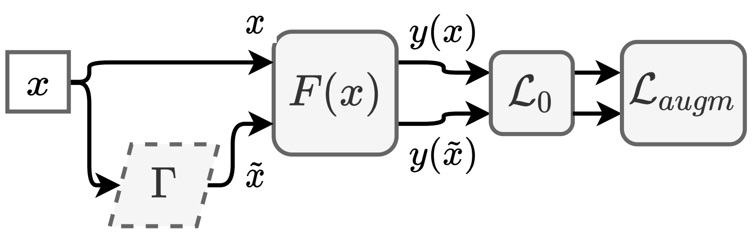
$\mathcal{D}$ is modeled as *Kullback–Leibler divergence* $\mathcal{D}_{KL}$, which measures the correspondence between the likelihood of the original and the perturbed input:

$$\mathcal{L}_{sim}(x, \tilde{x}; \theta) = \sum\nolimits_i \mathcal{D}_{KL}\big(R(x_i) \| Q(\tilde{x}_i)\big), \tag{3.10}$$

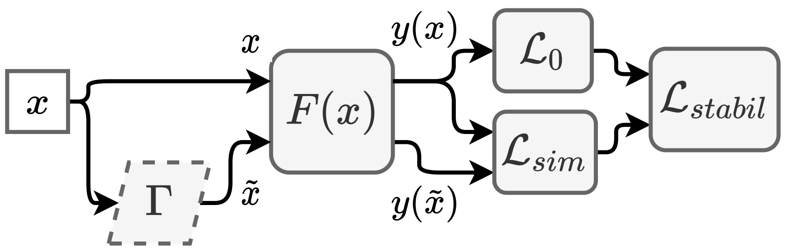
$$\mathcal{D}_{KL}\big(R(x) \| Q(\tilde{x})\big) = \sum\nolimits_j P(y_j|x) \log \frac{P(y_j|x)}{P(y_j|\tilde{x})}, \tag{3.11}$$

where i, j are the token and the class label indices, respectively. Figure 3.4b summarizes the main idea of the proposed stability training objective.

A critical difference between the data augmentation and the stability training objective is that the latter does not use noisy samples for the original task, but only for the stability objective. Both objectives could also be combined and used together, however, the goal of this work is to study their impact on robustness separately, and further exploration of the synergies between different training objectives is left to future work. Furthermore, both methods need perturbed



(a) Data Augmentation Training Objective $\mathcal{L}_{augm}$



(b) Stability Training Objective $\mathcal{L}_{stabil}$

Figure 3.4: Schema of the auxiliary training objectives. x, $\tilde{x}$ are the original, and the perturbed inputs, respectively, that are fed to the sequence labeling system $F(x)$. Γ represents a noising process. $y(x)$ and $y(\tilde{x})$ are the output distributions over the entity classes for x and $\tilde{x}$, respectively. $\mathcal{L}_0$ is the standard training objective. $\mathcal{L}_{augm}$ combines $\mathcal{L}_0$ computed on both outputs from $F(x)$. $\mathcal{L}_{stabil}$ fuses $\mathcal{L}_0$ calculated on the original input with the similarity objective $\mathcal{L}_{sim}$.

copies of the input samples, which results in longer training time but could be ameliorated by fine-tuning the existing model for a few epochs[8].

3.4 Experimental Setup

3.4.1 Data Sets and Tasks

Named Entity Recognition Task

The experiments that examine the proposed NAT approach are primarily focused on the NER scenario. To this end, the CoNLL 2003 (Tjong Kim Sang and De Meulder, 2003) and the GermEval 2014 (Benikova et al., 2014) data sets were employed. These data sets contain annotations for the following entity categories: person names (PER), organizations (ORG), locations (LOC), and miscellaneous entities (MISC). The detailed statistics of both data sets are presented in Tables B.1 and B.2 in the appendix.

[8] This setting was not explored in this thesis, leaving such optimization to future work.

Following Akbik et al. (2018), the revisited version of German CoNLL 2003 was used, which was prepared in 2006 and is believed to be more accurate, as the previous version was done by non-native speakers[9]. Moreover, only the inner layer of annotation for GermEval 2014 was employed in this work.

Syntactic Chunking and Part-of-Speech Tagging Tasks

In addition to the NER task, the utility of the presented approach is further validated in other sequence labeling scenarios:

- Syntactic chunking refers to the task of dividing the text into syntactically related, nonoverlapping groups of words (chunks).
- Part-of-speech tagging (POST) is the process of tagging each word in the text with the corresponding part of speech.

For the syntactic chunking experiment, the CoNLL 2000 data set (Tjong Kim Sang and Buchholz, 2000) was employed. To evaluate POST, both the English (UD English EWT[10]; Silveira et al., 2014) and the German (UD German GSD[11]) Universal Dependency Treebanks were exploited. The detailed statistics of these data sets are presented in Tables B.3, B.4 and B.6 in the appendix.

3.4.2 Model Architecture

In all experiments presented in this chapter, the BLSTM-CRF architecture (Huang et al., 2015) with a single BLSTM layer and 256 hidden units in both directions was used for $f(x)$. Moreover, four different text representations used to retrieve token embeddings $e(x)$ were considered. These representations were previously used to achieve state-of-the-art results on the studied data set and should also be able to handle misspelled text and out-of-vocabulary tokens:

- *FLAIR* (Akbik et al., 2018) learns a bidirectional language model using a BLSTM network to represent any sequence of characters. Following previous work, FLAIR was combined with GloVe embeddings (Pennington et al., 2014) for English and Wikipedia FastText word vectors (Bojanowski et al., 2017) for German. Moreover, FLAIR was also coupled with word vectors trained on Common Crawl (Mikolov et al., 2018) for both languages in the syntactic chunking and POST experiments.

[9] The revisited annotations are available on the official website of the CoNLL 2003 shared task: `https://www.clips.uantwerpen.be/conll2003/ner`.

[10] `https://universaldependencies.org/treebanks/en_ewt` (version 2.5)

[11] `https://universaldependencies.org/treebanks/de_gsd` (version 2.5)

- *BERT* (Devlin et al., 2019) uses a Transformer encoder to learn a bidirectional language model from large unlabeled text corpora and subword units to represent textual tokens. In the experiments presented in this chapter, the $\text{BERT}_{\text{BASE}}$ model was employed.
- *ELMo* (Peters et al., 2018) utilizes a linear combination of hidden state vectors derived from a BLSTM word language model trained on a large text corpus.
- *Glove/Wiki + Char* is a combination of pretrained word embeddings (GloVe for English and Wikipedia FastText for German) and randomly initialized character embeddings (Lample et al., 2016).

3.4.3 Training Setup

Two components of the sequence labeling architecture (Figure 3.2) were trained from scratch: (1) the sequence labeling model $f(x)$ and (2) the final CRF decoding layer. The pretrained embedding vectors $e(x)$ were fixed during training, except for the character embeddings. To train the NAT models, a mixture of the original data and its perturbed copies was used. The latter was generated from the synthetic noise distribution (Section 3.3.1) with the proposed noise induction procedure (Section 3.3.2).

Table A.3 in the appendix presents the detailed hyperparameters of the sequence labeling model $f(x)$ employed in the experiments described in this chapter. The hyperparameters were consistent with Akbik et al. (2018). All models were trained for at most 100 epochs. Moreover, *early stopping* was used based on the development set performance, measured as an average F_1 score of clean and noisy samples. Note that the development sets of each benchmark data set were used for validation only and not for training.

3.4.4 Evaluation Setup

The entity-level micro average F_1 score on the test set was used as the evaluation metric except for the POST task, where per token accuracy was used. The evaluation on both the original data sets and their copies perturbed using two common natural error distributions was performed as follows:

- **OCR errors**: The OCR errors were induced using the character confusion matrix Γ (Section 3.3.2) that was estimated based on the results of the Tesseract 3.04 OCR engine, which was employed to digitize the document images from the corpus detailed in Section 2.5.1. All characters that are not present in the original alphabet Σ of the corresponding data set for the downstream task were filtered out to avoid introducing out-of-alphabet symbols in the noise induction stage.

- **Misspellings**: Two sets of misspellings that were previously released by Belinkov and Bisk (2018) and Piktus et al. (2019) were employed in this scenario. Following the authors, every original token was replaced with the corresponding misspelled variant, sampling uniformly among available replacement candidates.

Table 3.1 presents the estimated error rates of text that is produced by executing these noise induction procedures on all employed test sets. Moreover, as the evaluation with noisy data leads to some variance in the final scores, all experiments were repeated five times, and the mean and standard deviation values were reported.

Table 3.1: Estimated error rates of text produced using different noise distributions (Section 3.4.4). OCR noise is modeled with the character confusion matrix, whereas misspellings are induced using lookup tables released by Belinkov and Bisk (2018)† and Piktus et al. (2019)‡.

(a) Character Error Rates

Data Set	OCR Noise	Misspellings†	Misspellings‡
English CoNLL 2003	8.9%	16.5%	9.8%
German CoNLL 2003	9.0%	8.3%	8.0%
GermEval 2014	9.3%	8.6%	8.2%
CoNLL 2000	8.5%	12.4%	9.0%
UD English EWT	8.9%	13.9%	9.2%
UD German GSD	8.7%	9.8%	8.7%

(b) Word Error Rates

Data Set	OCR Noise	Misspellings†	Misspellings‡
English CoNLL 2003	35.6%	55.4%	48.3%
German CoNLL 2003	39.5%	26.5%	45.5%
GermEval 2014	41.2%	27.0%	47.9%
CoNLL 2000	35.6%	45.9%	46.3%
UD English EWT	34.4%	48.4%	44.8%
UD German GSD	38.9%	30.5%	50.4%

3.4.5 Implementation Details

All models were implemented using the FLAIR framework (Akbik, Bergmann, Blythe, et al., 2019)[12]. The basic sequence labeling model in this framework was extended by integrating the proposed auxiliary training objectives (Sections 3.3.3 and 3.3.4). Nonetheless, the approach presented in this chapter is universal and can be implemented in any other sequence labeling framework.

3.5 Experimental Results

3.5.1 Named Entity Recognition

Firstly, the NAT approach was validated in the NER scenario. To this end, the baseline models were trained with and without the proposed auxiliary loss objectives (Sections 3.3.3 and 3.3.4). The CoNLL 2003 and the GermEval 2014 data sets were employed (Section 3.4.1), and a label-preserving training setup was used in this experiment, i.e., $\alpha = 1.0$ and $\eta_{train} = 10\%$.

The baselines utilized GloVe vectors coupled with FLAIR and character embeddings (*FLAIR + GloVe* and *GloVe + Char*, respectively), *BERT*, and *ELMo* embeddings for English. For German, Wikipedia FastText vectors paired with FLAIR and character embeddings (*FLAIR + Wiki* and *Wiki + Char*, respectively) were employed.[13]

Tables 3.2 to 3.4 present the results of this experiment[14]. The proposed auxiliary training objectives significantly improved accuracy on noisy input data containing OCR errors and misspelling for all baseline models and both languages. At the same time, they preserved the accuracy for the original input. Moreover, the data augmentation objective performed slightly better than the stability training objective. However, the chosen hyperparameter values were rather arbitrary, as the goal of this experiment was to prove the utility and the flexibility of both objectives. Therefore, the achieved results can most likely be improved.

[12] FLAIR v0.4.2; `https://github.com/flairNLP/flair`

[13] This choice was motivated by the availability of pretrained embedding models in the FLAIR framework.

[14] The exact results from the original papers could not be replicated because the development sets were not used for training in this work. Moreover, the presented approach is feature-based, as the embeddings are not fine-tuned on the target task.

Table 3.2: Evaluation results of NER on the English CoNLL 2003 test set (Table B.1a). The original data, as well as its noisy copies with OCR errors and two types of misspellings released by Belinkov and Bisk (2018)† and Piktus et al. (2019)‡ were used. $\mathcal{L}_0$, $\mathcal{L}_{augm}$, and $\mathcal{L}_{stabil}$ are the standard, the data augmentation, and the stability objectives, respectively. Mean F$_1$ scores with standard deviations from five experiments and mean differences against the standard objective (in parentheses) are reported.

(a) FLAIR + GloVe Embeddings

Train loss	Original data	OCR errors	Misspellings†	Misspellings‡
$\mathcal{L}_0$	92.05	76.44±0.45	75.09±0.48	87.57±0.10
$\mathcal{L}_{augm}$	**92.56** (+0.51)	**84.79±0.23** (+8.35)	**83.57±0.43** (+8.48)	**90.50±0.08** (+2.93)
$\mathcal{L}_{stabil}$	91.99 (-0.06)	84.39±0.37 (+7.95)	82.43±0.23 (+7.34)	90.19±0.14 (+2.62)

(b) BERT Embeddings

Train loss	Original data	OCR errors	Misspellings†	Misspellings‡
$\mathcal{L}_0$	90.91	68.23±0.39	65.65±0.31	85.07±0.15
$\mathcal{L}_{augm}$	90.84 (-0.07)	**79.34±0.32** (+11.11)	**75.44±0.28** (+9.79)	86.21±0.24 (+1.14)
$\mathcal{L}_{stabil}$	**90.95** (+0.04)	78.22±0.17 (+9.99)	73.46±0.34 (+7.81)	**86.52±0.12** (+1.45)

(c) ELMo Embeddings

Train loss	Original data	OCR errors	Misspellings†	Misspellings‡
$\mathcal{L}_0$	**92.16**	72.90±0.50	70.99±0.17	88.59±0.19
$\mathcal{L}_{augm}$	91.85 (-0.31)	**84.09±0.18** (+11.19)	**82.33±0.40** (+11.34)	**89.50±0.16** (+0.91)
$\mathcal{L}_{stabil}$	91.78 (-0.38)	83.86±0.11 (+10.96)	81.47±0.29 (+10.48)	89.49±0.15 (+0.90)

(d) GloVe + Char Embeddings

Train loss	Original data	OCR errors	Misspellings†	Misspellings‡
$\mathcal{L}_0$	90.26	71.15±0.51	70.91±0.39	87.14±0.07
$\mathcal{L}_{augm}$	**90.83** (+0.57)	**81.09±0.47** (+9.94)	**79.47±0.24** (+8.56)	**88.82±0.06** (+1.68)
$\mathcal{L}_{stabil}$	90.21 (-0.05)	80.33±0.29 (+9.18)	78.07±0.23 (+7.16)	88.47±0.13 (+1.33)

Table 3.3: Evaluation results of NER on the German CoNLL 2003 test set (Table B.1b). The original data, as well as its noisy copies with OCR errors and two types of misspellings released by Belinkov and Bisk (2018)† and Piktus et al. (2019)‡ were used. $\mathcal{L}_0$, $\mathcal{L}_{augm}$, and $\mathcal{L}_{stabil}$ are the standard, the data augmentation, and the stability objectives, respectively. Mean F_1 scores with standard deviations from five experiments and mean differences against the standard objective (in parentheses) are reported.

(a) FLAIR + Wiki

Train loss	Original data	OCR errors	Misspellings†	Misspellings‡
$\mathcal{L}_0$	86.13	66.93±0.49	78.06±0.13	80.72±0.23
$\mathcal{L}_{augm}$	**86.46** (+0.33)	**75.90±0.63** (+8.97)	**83.23±0.14** (+5.17)	84.01±0.27 (+3.29)
$\mathcal{L}_{stabil}$	86.33 (+0.20)	75.08±0.29 (+8.15)	82.60±0.21 (+4.54)	**84.12±0.26** (+3.40)

(b) Wiki + Char

Train loss	Original data	OCR errors	Misspellings†	Misspellings‡
$\mathcal{L}_0$	82.20	59.15±0.76	75.27±0.31	71.45±0.15
$\mathcal{L}_{augm}$	**82.62** (+0.42)	67.67±0.75 (+8.52)	**78.48±0.24** (+3.21)	79.14±0.31 (+7.69)
$\mathcal{L}_{stabil}$	82.18 (-0.02)	**67.72±0.63** (+8.57)	77.59±0.12 (+2.32)	**79.33±0.39** (+7.88)

Table 3.4: Evaluation results of NER on the GermEval 2014 test set (Table B.2). The original data, as well as its noisy copies with OCR errors and two types of misspellings released by Belinkov and Bisk (2018)† and Piktus et al. (2019)‡ were used. $\mathcal{L}_0$, $\mathcal{L}_{augm}$, and $\mathcal{L}_{stabil}$ are the standard, the data augmentation, and the stability objectives, respectively. Mean F_1 scores with standard deviations from five experiments and mean differences against the standard objective (in parentheses) are reported.

(a) FLAIR + Wiki

Train loss	Original data	OCR errors	Misspellings†	Misspellings‡
$\mathcal{L}_0$	**85.05**	58.64±0.51	67.96±0.23	68.64±0.28
$\mathcal{L}_{augm}$	84.84 (-0.21)	**72.02±0.24** (+13.38)	**78.59±0.11** (+10.63)	**81.55±0.12** (+12.91)
$\mathcal{L}_{stabil}$	84.43 (-0.62)	70.15±0.27 (+11.51)	75.67±0.16 (+7.71)	79.31±0.32 (+10.67)

(b) Wiki + Char

Train loss	Original data	OCR errors	Misspellings†	Misspellings‡
$\mathcal{L}_0$	80.32	52.48±0.31	61.99±0.35	54.86±0.15
$\mathcal{L}_{augm}$	**80.68** (+0.36)	**63.74±0.31** (+11.26)	**70.83±0.09** (+8.84)	**75.66±0.11** (+20.80)
$\mathcal{L}_{stabil}$	80.00 (-0.32)	62.29±0.35 (+9.81)	68.23±0.23 (+6.24)	72.40±0.29 (+17.54)

3.5.2 Syntactic Chunking and Part-of-Speech Tagging

Secondly, to confirm that the NAT approach is applicable beyond the NER task, the NER experiment (Section 3.5.1) was replicated in other sequence labeling scenarios, i.e., syntactic chunking and POST (Section 3.4.1). A label-preserving training setup was used, i.e., $\alpha = 1.0$ and $\eta_{train} = 10\%$, except that the results on the UD English EWT and the CoNLL 2000 data set with misspellings were obtained using a lower sampling rate of 70%.

The syntactic chunking experiment was performed on the CoNLL 2000 data set. The baseline models utilized ELMo embeddings and FLAIR embeddings coupled with word vectors trained on Common Crawl (Mikolov et al., 2018), referred to as *ELMo* and *FLAIR + Crawl* embeddings, respectively.

In the POST experiment on the UD English EWT data set, FLAIR embeddings paired with GloVe and Common Crawl word vectors (*FLAIR + GloVe* and *FLAIR + Crawl*, respectively) were employed. For experiments on the UD German GSD data set, FLAIR embeddings coupled with Wikipedia FastText and Common Crawl word vectors (*FLAIR + Wiki* and *FLAIR + Crawl*, respectively) were employed.

The results of the syntactic chunking and POST experiments are presented in Tables 3.5 to 3.7. They confirm that NAT improves the accuracy of sequence labeling beyond the recognition of entities. The observed improvements are even more pronounced than in the case of NER. Apparently, this is due to the fact that entities are relatively rare in natural text. In contrast, the annotations in the case of the syntactic chunking and the POST tasks are much denser, which could explain the observed difference in the level of improvements between the tasks.

3.5.3 Sensitivity Analysis

This experiment aims to evaluate the impact of the hyperparameters of the proposed method on sequence labeling accuracy. In this scenario, the English CoNLL 2003 data set was used, and multiple models were trained with different amounts of synthetic noise that was induced at training time (η_{train}) and different weighting factors of the auxiliary training objectives (α). For comparison, the models that employed the test-time error distribution at training time were also examined. The *FLAIR + GloVe* model was chosen as a baseline because it achieved the best results in the preliminary analysis (Table 3.2) and performed satisfactorily in terms of the execution time, which allowed performing extensive experiments.

Table 3.5: Evaluation results of syntactic chunking on the CoNLL 2000 data set (Table B.3). The original data, as well as its noisy copies with OCR errors and two types of misspellings released by Belinkov and Bisk (2018)† and Piktus et al. (2019)‡ were used. $\mathcal{L}_0$, $\mathcal{L}_{augm}$, and $\mathcal{L}_{stabil}$ are the standard, the data augmentation, and the stability objectives, respectively. Mean F_1 scores with standard deviations from five experiments and mean differences against $\mathcal{L}_0$ (in parentheses) are reported.

(a) FLAIR + Crawl Embeddings

Train loss	Original data	OCR errors	Misspellings†	Misspellings‡
$\mathcal{L}_0$	**96.39**	80.55±0.11	61.87±0.39	64.23±0.25
$\mathcal{L}_{augm}$	96.37 (-0.02)	**91.44±0.15** (+10.89)	**81.89±0.23** (+20.02)	**85.97±0.34** (+21.74)
$\mathcal{L}_{stabil}$	96.27 (-0.12)	90.45±0.12 (+9.90)	80.81±0.17 (+18.94)	84.16±0.27 (+19.93)

(b) ELMo Embeddings

Train loss	Original data	OCR errors	Misspellings†	Misspellings‡
$\mathcal{L}_0$	96.56	80.55±0.21	61.41±0.24	62.65±0.48
$\mathcal{L}_{augm}$	**96.64** (+0.08)	**90.92±0.18** (+10.37)	**80.04±0.34** (+18.63)	**84.11±0.19** (+21.46)
$\mathcal{L}_{stabil}$	96.50 (-0.06)	89.87±0.22 (+9.32)	78.97±0.17 (+17.56)	82.29±0.18 (+19.64)

Table 3.6: Evaluation results of POST on the UD English EWT test set (Table B.4). The original data, as well as its noisy copies with OCR errors and two types of misspellings released by Belinkov and Bisk (2018)† and Piktus et al. (2019)‡ were used. $\mathcal{L}_0$, $\mathcal{L}_{augm}$, and $\mathcal{L}_{stabil}$ are the standard, the data augmentation, and the stability training objectives, respectively. Per token accuracy with standard deviations from five experiments and mean differences against $\mathcal{L}_0$ (in parentheses) are reported.

(a) FLAIR + GloVe Embeddings

Train loss	Original data	OCR errors	Misspellings†	Misspellings‡
$\mathcal{L}_0$	**94.03**	72.03±0.30	52.13±0.39	58.41±0.22
$\mathcal{L}_{augm}$	93.49 (-0.54)	83.17±0.30 (+11.14)	68.29±0.36 (+16.16)	75.23±0.14 (+16.82)
$\mathcal{L}_{stabil}$	93.61 (-0.42)	**83.62±0.36** (+11.59)	**69.23±0.34** (+17.10)	**75.91±0.24** (+17.50)

(b) FLAIR + Crawl Embeddings

Train loss	Original data	OCR errors	Misspellings†	Misspellings‡
$\mathcal{L}_0$	94.02	71.99±0.37	54.01±0.49	59.24±0.25
$\mathcal{L}_{augm}$	93.89 (-0.13)	**84.14±0.36** (+12.15)	70.68±0.21 (+16.67)	**76.97±0.31** (+17.73)
$\mathcal{L}_{stabil}$	**94.04** (+0.02)	84.09±0.23 (+12.10)	**71.00±0.27** (+16.99)	76.95±0.21 (+17.71)

Table 3.7: Evaluation results of POST on the UD German GSD test set (Table B.6). The original data, as well as its noisy copies with OCR errors and two types of misspellings released by Belinkov and Bisk (2018)† and Piktus et al. (2019)‡ were used. $\mathcal{L}_0$, $\mathcal{L}_{augm}$, and $\mathcal{L}_{stabil}$ are the standard, the data augmentation, and the stability training objectives, respectively. Per token accuracy with standard deviations from five experiments and mean differences against $\mathcal{L}_0$ (in parentheses) are reported.

(a) FLAIR + Wiki Embeddings

Train loss	Original data	OCR errors	Misspellings†	Misspellings‡
$\mathcal{L}_0$	90.24	68.35±0.37	64.81±0.19	49.85±0.30
$\mathcal{L}_{augm}$	**90.30** (+0.06)	**84.17±0.41** (+15.82)	**75.25±0.27** (+10.44)	**77.61±0.39** (+27.76)
$\mathcal{L}_{stabil}$	90.08 (-0.16)	83.27±0.23 (+14.92)	74.80±0.24 (+9.99)	76.82±0.32 (+26.97)

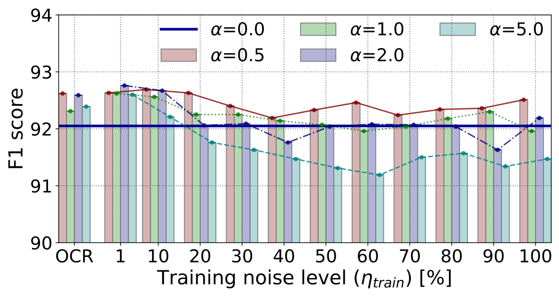
(b) FLAIR + Crawl Embeddings

Train loss	Original data	OCR errors	Misspellings†	Misspellings‡
$\mathcal{L}_0$	90.10	66.17±0.34	63.19±0.22	49.05±0.42
$\mathcal{L}_{augm}$	**90.12** (+0.02)	**84.13±0.32** (+17.96)	**74.57±0.23** (+11.38)	**78.16±0.17** (+29.11)
$\mathcal{L}_{stabil}$	89.97 (-0.13)	83.11±0.35 (+16.94)	74.27±0.15 (+11.08)	77.27±0.35 (+28.22)

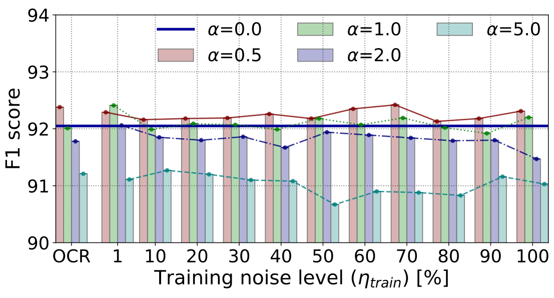
Figure 3.5 summarizes the results of the sensitivity experiment. In most cases, the models trained with the proposed auxiliary NAT objectives preserved or even improved the accuracy on the original test data compared to the baseline model ($\alpha = 0$). Moreover, they significantly outperformed the baseline model on test data perturbed with OCR noise. The highest accuracy was achieved for η_{train} from 10 to 30%, which roughly corresponds to the label-preserving noise range (Section 3.2.3).

The best choice of α was in the range from 0.5 to 2.0. In the case that the weight of the auxiliary training objective is set to $\alpha = 5.0$, we observe lower accuracy on the original, error-free data. Furthermore, the models trained on the real error distribution demonstrated at most slightly better performance, which indicates that the exact noise distribution might not necessarily have to be known at training time. Note that the aspect of mimicking an empirical noise distribution is further explored in Chapter 4.

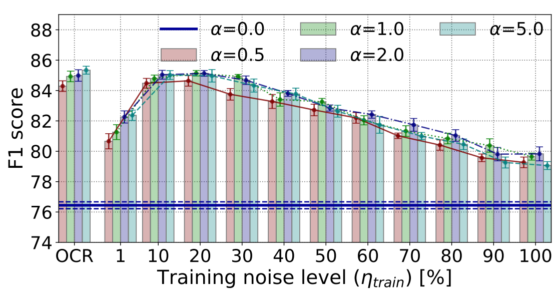
Similar to Heigold et al. (2018) and Cheng et al. (2019), it can be concluded that a nonzero noise level induced during training ($\eta_{train} > 0$) always yields improvements on noisy input data when compared with the models trained exclusively on error-free examples.



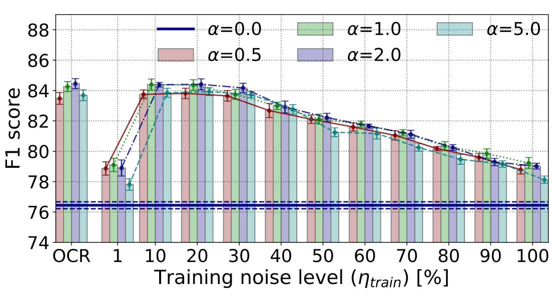
(a) Data augmentation (original test data)



(b) Stability training (original test data)



(c) Data augmentation (tested on OCR errors)



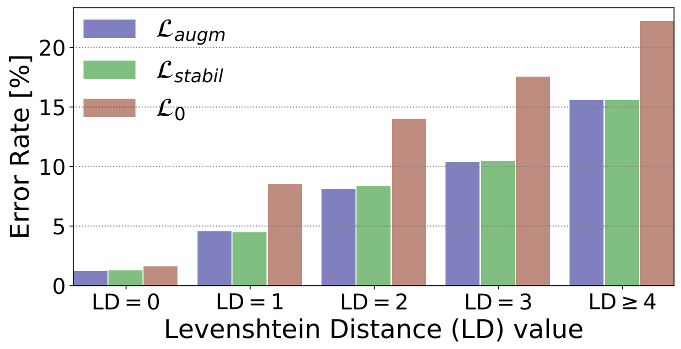
(d) Stability training (tested on OCR errors)

Figure 3.5: Sensitivity analysis performed on the English CoNLL 2003 test set (Table B.1a). Each figure presents the results of a model trained using one of the proposed auxiliary training objectives (Sections 3.3.3 and 3.3.4) and evaluated on either original data or its variant perturbed with OCR errors (Section 3.4.4). The bar marked as *OCR* represents a model trained using the test-time OCR noise distribution. Other bars correspond to models trained using synthetic noise with different noising rates η_{train} and weighting factors α.

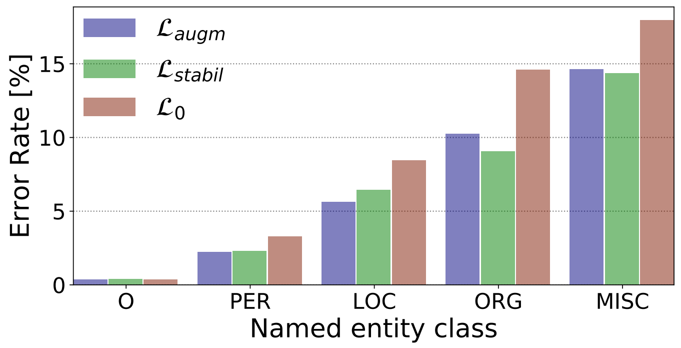
3.5.4 Error Analysis

To further quantify the improvements provided by the proposed approach, as well as to identify its potential drawbacks and limitations, the accuracy of sequence labeling was measured on the subsets of data with different levels of perturbation. More specifically, the input tokens were grouped based on edit distance to their error-free counterparts. Moreover, the data was partitioned by the named entity class to assess the impact of noise on the recognition of different entity types.

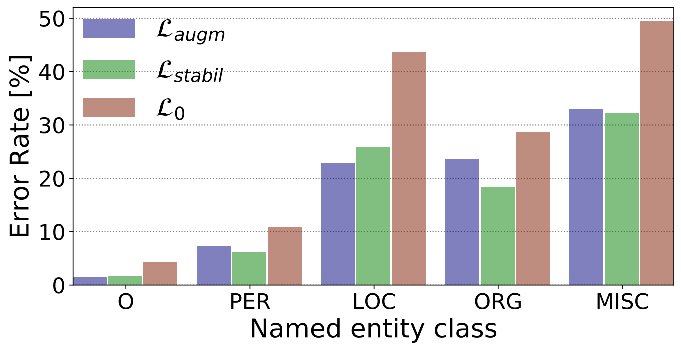
For this experiment, both the test and the development parts of the English CoNLL 2003 data set were jointly used as the evaluation data set. To imitate the noisy input, OCR errors were induced to the original, error-free data as described in Section 3.4.4. Subsequently, the subset-level error rates were calculated for the models that employed either the standard or the proposed auxiliary training objectives. Figure 3.6 presents the results of this experiment.



(a) Data divided by the edit distance value



(b) Data divided by the entity class (error-free tokens)



(c) Data divided by the entity class (perturbed tokens)

Figure 3.6: Error analysis on the English CoNLL 2003 data set with OCR noise. The results of the *FLAIR + GloVe* model trained with the standard, the data augmentation, and the stability training objectives ($\mathcal{L}_0$, $\mathcal{L}_{augm}$, and $\mathcal{L}_{stabil}$, respectively) are presented. The data was divided into subsets based on the edit distance of a token to its original counterpart and its named entity class. The latter group was further partitioned into the error-free and the perturbed tokens. The error rate is the percentage of tokens with misrecognized entity class labels.

It can be seen that the NAT approach achieved significant error reduction across all perturbation levels and all entity types. Moreover, narrowing down the analysis to perturbed tokens reveals that the baseline model was particularly sensitive to noisy tokens from the LOC and the MISC categories. The proposed NAT approach considerably reduced this negative effect.

Furthermore, as the stability training worked slightly better on the LOC class and the data augmentation was more accurate on the ORG type, both methods could possibly be combined to enhance overall sequence labeling accuracy further. Finally, note that even if the particular token was not perturbed, its context could be noisy, which would explain the fact that the proposed approach provided improvements even for tokens without perturbations.

3.5.5 Qualitative Analysis

In this section, the outputs of the models trained with and without the NAT method generated on erroneous input text are qualitatively compared. The results in Figures 3.7 and 3.8 show that the proposed method improved robustness to:

(1) Capitalization errors — see the 1st and the 5th example in Figure 3.8.

(2) Substitutions of characters — see the 2nd example in Figure 3.8 as well as the 1st, the 2nd, the 4th, and the 5th example in Figure 3.7.

(3) Deletions of characters — see the 3rd and the 6th example in Figure 3.8.

(4) Insertions of characters — see the 3rd and the 5th example in Figure 3.7.

Moreover, the proposed method better recognized the semantics of the sentence in the 4th row of Figure 3.8, where the location name was creatively rewritten (*Brazland* instead of *Brazil*). Furthermore, note that misrecognition could also be caused by the errors in the context words. In the 7th example in Figure 3.8, the organization name *Bundesliga* is spelled correctly, but the preceding words are misspelled, which apparently confused the model trained using the standard objective and caused the model to omit this entity. In contrast, the model that employed the NAT method correctly identified all entities in this example.

3.6 Related Work

This section presents the most related approaches for improving robustness to nonstandard or erroneous input (Sections 3.6.2 to 3.6.4), as well as the prior work on the impact of noisy input data on the downstream task performance (Section 3.6.1).

Reference #1	`Hapoel Jerusalem 12 4 1 7 10 18 13`
Noisy Reference	`Hapoel lerusalem I2 A 1 7 10 18 13`
NAT Model	`Hapoel lerusalem I2 A 1 7 10 18 13`
Standard Model	`Hapoel lerusalem I2 A 1 7 10 18 13`
Reference #2	`SOCCER - SPANISH FIRST DIVISION RESULT / STANDINGS.`
Noisy Reference	`SOCCER - SPAN1SH FIRST DIVISiOW RESULT / STA'DINGS.`
NAT Model	`SOCCER - SPAN1SH FIRST DIVISiOW RESULT / STA'DINGS.`
Standard Model	`SOCCER - SPAN1SH FIRST DIVISiOW RESULT / STA'DINGS.`
Reference #3	`EU , Poland agree on oil import tariffs.`
Noisy Reference	`EU , Po'land agree on oil import tarifs.`
NAT Model	`EU , Po'land agree on oil import tarifs.`
Standard Model	`EU , Po'land agree on oil import tarifs.`
Reference #4	`Schlamm scheint zu helfen - Yahoo!`
Noisy Reference	`Schlamm scheint zu helfen - Yaho0!`
NAT Model	`Schlamm scheint zu helfen - Yaho0!`
Standard Model	`Schlamm scheint zu helfen - Yaho0 !`
Reference #5	`Fachverband für Hauswirtschaft :`
Noisy Reference	`Fachverbandi für Hauswi'tschaTt :`
NAT Model	`Fachverbandi für Hauswi'tschaTt :`
Standard Model	`Fachverbandi für Hauswi'tschaTt :`

Figure 3.7: Outputs produced by the models trained either with the standard or the auxiliary NAT objectives on the input text that contains OCR errors. The examples demonstrate the cases where the models trained with the NAT objectives correctly recognized all tags, while the baseline models either misclassified or completely missed some entities. The background color indicates the entity type: person name (blue), location (green), organization (orange), or miscellaneous entity (gray). Best viewed in color.

Reference #1	`7-1 Raul Gonzalez 7-1 Juan Pizzi`
Noisy Reference	`7-1 raul gonzalez 7-1 juan Pizzi`
NAT Model	`7-1 raul gonzalez 7-1 juan Pizzi`
Standard Model	`7-1 raul gonzalez 7-1 juan Pizzi`
Reference #2	`6. Heidi Zurbriggen (Switzerland) 153`
Noisy Reference	`6. Heidi Zurbriggen (swizzerland) 153`
NAT Model	`6. Heidi Zurbriggen (swizzerland) 153`
Standard Model	`6. Heidi Zurbriggen (swizzerland) 153`
Reference #3	`Damascus denies aiding the rebels.`
Noisy Reference	`Damascuse denies aiding de rebels.`
NAT Model	`Damascuse denies aiding de rebels.`
Standard Model	`Damascuse denies aiding de rebels.`
Reference #4	`Plastic surgery gets boost in Brazil.`
Noisy Reference	`Plastic surgury hets boost is Brazland.`
NAT Model	`Plastic surgury hets boost is Brazland.`
Standard Model	`Plastic surgury hets boost is Brazland.`
Reference #5	`Waltraud Zimmer, Rödermark-Ober-Roden`
Noisy Reference	`Waltraud zimmer, Rödermark-Ober-Roden`
NAT Model	`Waltraud zimmer, Rödermark-Ober-Roden`
Standard Model	`Waltraud zimmer, Rödermark-Ober-Roden`
Reference #6	`Deutschland ist noch nicht Teil der Reiseroute."`
Noisy Reference	`Deutshland is nach nich Teil der Reiseroute."`
NAT Model	`Deutshland is nach nich Teil der Reiseroute."`
Standard Model	`Deutshland is nach nich Teil der Reiseroute."`
Reference #7	`Auch für sie kostet die Bundesliga 14,90 Euro im Monat.`
Noisy Reference	`Aauch fur si kosstet di Bundesliga 14,90 Euro im Monat.`
NAT Model	`Aauch fur si kosstet di Bundesliga 14,90 Euro im Monat.`
Standard Model	`Aauch fur si kosstet di Bundesliga 14,90 Euro im Monat.`

Figure 3.8: Outputs produced by the models trained either with the standard or the auxiliary NAT objectives on the input text that contains misspellings. The examples demonstrate the cases where the models trained with the NAT objectives correctly recognized all tags, while the baseline models either misclassified or completely missed some entities. The background color indicates the entity type: person name (blue), location (green), organization (orange), or miscellaneous entity (gray). Best viewed in color.

3.6.1 The Impact of Noisy Input Data

Errors of OCR, ASR, and other text generators always pose a challenge to the downstream NLP systems (Alex and Burns, 2014; Lopresti, 2009; Packer et al., 2010; Ruiz et al., 2017).

Previous research concerning the impact of noisy input on downstream task performance was primarily carried out in the context of ASR and NMT. Ruiz et al. (2017) noted that machine translation systems are conventionally trained on the text that does not exhibit phenomena that occur in spoken language. They employed state-of-the-art machine translation systems to translate utterances that contain ASR errors and identified new research areas in evaluating NMT systems for spoken language translation. Chen et al. (2017) investigated the effects of ASR errors on spoken dialog systems and noted that the basic approach of cascading ASR modules with the downstream text processing systems works well only when the ASR accuracy is relatively high. X. Li et al. (2018) stated that recent NMT systems trained using error-free parallel data sets cannot properly handle sentences produced by the ASR system due to erroneous input.

In the document processing field, Lopresti (2009) investigated the impact of noise that is induced by OCR and stated that text recognition errors present a serious challenge to the downstream NLP systems that make use of such data. Moreover, Packer et al. (2010) studied the task of extracting person names from digitized historical documents and argued that accurate identification of named entities in the noisy output of an OCR engine presents a challenge beyond previously explored NER scenarios. Furthermore, Alex and Burns (2014) analyzed NER performed on several digitized historical text collections and showed that OCR errors have a significant impact on the accuracy of the downstream task.

Although the OCR technology is more advanced than several years ago when many historical archives were digitized (Kim and Cassidy, 2015; Neudecker, 2016), recall that in Chapter 2, the efficiency of modern OCR engines was examined, and it was shown that the most widely used methods still have difficulties with nonstandard or lower quality input documents.

3.6.2 Noise-Additive Data Augmentation

A widely adopted method of improving robustness to nonstandard input is to augment the training data with examples perturbed using a model that mimics the error distribution to be encountered at test time (Cubuk et al., 2019; Krizhevsky et al., 2012; Lim et al., 2019; Tsvetkov et al., 2014).

Apparently, the exact modeling of noise might be impractical or even impossible, thus, methods that employ randomized error patterns for training recently gained increasing popularity (Belinkov and Bisk, 2018; Heigold et al., 2018; Karpukhin et al., 2019; Lakshmi Narayan et al., 2019; Sperber et al., 2017). Although trained using synthetic errors, these methods are often able to achieve moderate improvements on data from natural sources of noise.

Sperber et al. (2017) randomly induced errors to the source side of parallel training data for speech translation and achieved moderately improved robustness with properly calibrated type and amount of noise that is induced at training time. Heigold et al. (2018) demonstrated that the noisy input substantially degrades the accuracy of models trained on error-free data. They used word scrambling, as well as character flips and swaps as their noise model, and achieved the best results under matched training and test noise conditions.

Belinkov and Bisk (2018) reported significant degradation in the performance of NMT systems on noisy input. They built a lookup table of possible lexical replacements from Wikipedia edit histories and used it as a natural source of the noise. Robustness to noise was only achieved by training with the same distribution — at the expense of performance degradation on other types of noise. In contrast, the method proposed in this chapter performed well on natural noise at test time by using a simplified synthetic noise model during training.

Noteworthy, Karpukhin et al. (2019) pointed out that existing NMT approaches are very sensitive to spelling mistakes and proposed to augment training samples with random character deletions, insertions, substitutions, and swaps. They showed improved robustness to natural noise, represented by frequent corrections in Wikipedia edit logs, without diminishing performance on the original data. However, not every word in the vocabulary has a corresponding misspelling. Therefore, even when noise is applied at the maximum rate, only a subset of tokens is perturbed (20-50%, depending on the language). In contrast, the approach presented in this chapter uses a confusion matrix, which is better suited to model statistical error distribution and can be applied to all tokens, not only those present in the corresponding lookup tables.

3.6.3 Noise-Invariant Latent Representation

Robustness can also be improved by designing a representation that is less sensitive to noise, in particular, by encouraging the models to learn a similar latent representation for both the error-free and the erroneous input.

Inspired by *contractive autoencoders* (Rifai et al., 2011), Y. Li et al. (2016) proposed to stabilize predictions by minimizing the ability of features to perturb results by keeping the variation of the output lower than the noise. They trained their models using first-order derivatives of the training loss as a part of the regularization term and used a word-level dropout as their noise model.

S. Zheng et al. (2016) introduced *stability training* — a general method used to stabilize predictions against small input perturbations. Cheng et al. (2018) continued their work and developed the adversarial stability training method for NMT by adding a discriminator term to the objective function. They induced noise by randomly replacing words in the input sentence or by adding Gaussian noise to word embeddings and evaluated their approach using random swaps, deletions, and replacements of words. They combined data augmentation and stability objectives, while in this work, both methods were evaluated separately and the evaluation results on natural noise distribution were provided.

Piktus et al. (2019) learned a representation that embeds misspelled words close to their correct variants. Their *misspelling oblivious embeddings* model jointly optimizes two loss functions, each of which iterates over a separate data set (a corpus of text and a set of misspelling/correction pairs) during training. In contrast, the method proposed in this chapter does not depend on any additional resources and uses a simplified error distribution during training.

Jia et al. (2019) focused on adversarial word substitutions and applied the *interval bound propagation* technique to minimize an upper bound on the worst-case loss for allowed word substitutions. Their method outperformed data augmentation on text classification data sets but also caused a moderate drop in the accuracy on unperturbed text compared with normal training. Moreover, it is still an open problem whether this model could handle other types of perturbations like insertions or deletions.

Jones et al. (2020) developed *robust encodings* that balance stability (consistent predictions across various perturbations) and fidelity (accuracy on unperturbed input) by mapping sentences to a smaller discrete space of encodings. Although their model improved robustness against small perturbations, it decreased accuracy on the error-free input.

Contemporary to the NAT method, Q. Xie et al. (2020) proposed the *unsupervised data augmentation* approach that encourages the predictions to be consistent between the original and the augmented examples from an unlabeled data set. Similar to the stability training objective, their method minimizes the Kullback–Leibler divergence between the predictions on the original and the augmented examples. Moreover, they proposed to induce perturbation using targeted data augmentation methods, i.e., *backtranslation* (Sennrich et al., 2016) for textual data and *AutoAugment* (Cubuk et al., 2019) for the input in image format. Their method leads to substantial improvements in several language and vision tasks.

3.6.4 Adversarial Learning

Adversarial examples are the inputs designed to mislead ML models (Goodfellow et al., 2015; Szegedy et al., 2014). Adversarial attacks aim to slightly modify the original input, so that the model misclassifies it, although the original example is classified correctly. In a white-box attack scenario (Ebrahimi et al., 2018) we assume that the attacker has access to the model parameters, in contrast to the black-box scenario (J. Gao et al., 2018), where the attacker can only sample model predictions on given input examples.

Many different methods were used to generate adversarial examples. Bekoulis et al. (2018), Miyato et al. (2017), and Yasunaga et al. (2018) added the worst-case perturbation of a small, bounded norm at the level of dense word and character embeddings. Ebrahimi et al. (2018) used the gradient of each perturbation to guide the sample selection process. Alzantot et al. (2018) exploited population-based, gradient-free optimization via genetic algorithms to generate natural language adversarial examples. H. Zhang et al. (2019) proposed the Metropolis-Hastings attack algorithm to improve the fluency of generated results.

On the other hand, *adversarial training* (Bekoulis et al., 2018; Miyato et al., 2017; Yasunaga et al., 2018) is a method that augments the training data with adversarial examples to make the ML models more resistant to adversarial input. The NAT method proposed in this chapter can be thought of as a black-box adversarial training method that utilizes naturally occurring adversarial examples in the form of noisy digitized or misspelled text.

3.7 Summary

In this chapter, we investigated the difference in accuracy between sequence labeling performed on error-free and noisy text (Section 3.2.3). To this end, the noisy sequence labeling problem was formulated, where the input may undergo an unknown noising process (Section 3.2.2). Moreover, in Section 3.3.1, the confusion matrix-based model was introduced. It can be used to either estimate the real noise distribution encountered in natural language text or to imitate mistyped or misrecognized text by using a simplified, uniform error distribution. Furthermore, the noise induction procedure was developed and used to simulate the real noisy text (Section 3.3.2).

Most importantly, two NAT objectives that boost sequence labeling accuracy on the perturbed text were proposed:

(1) The data augmentation approach uses a mixture of clean and noisy examples during training to make the model resistant to erroneous input (Section 3.3.3).

(2) The stability training algorithm encourages output similarity for the original and the perturbed input, which helps the model build a noise invariant latent representation (Section 3.3.4).

The performed experiments confirmed that NAT consistently improves the efficiency of popular sequence labeling models on data perturbed with different error distributions, preserving accuracy on the original input (Section 3.5). Moreover, the proposed method avoids expensive retraining of embeddings on noisy data sources by employing existing text representations.

The NAT approach makes existing models applicable beyond the idealized conditions for which they were originally designed. Although this method was not intended to correct erroneous text, it may support an automatic correction method that uses recognized entity types to narrow the list of feasible correction candidates. Another application is data anonymization (Biesner et al., 2022; Mamede et al., 2016), where only the locations in text and not the exact values of entities are required, as they will be anonymized anyway.

Future Work Directions

Future work will involve improvements in the proposed noise model to study the importance of fidelity to real-world error patterns.[15] Moreover, the evaluation of the NAT approach in handling errors from other real noise distributions (e.g., from ASR) and other sequence labeling tasks would further support the utility of the proposed method. Furthermore, NAT could be combined with techniques that allow for avoiding erroneous tokenization, e.g., the method proposed by Liu et al. (2019), which is orthogonal to the proposed approach.

[15] In Chapter 4, an improved error model that better mimics empirical error distribution is presented.

4 Empirical Error Modeling for Improved Noise-Aware Training

Preface

This chapter is adapted from Namysl, Behnke, and Köhler (2021)[1], previously published by the Association for Computational Linguistics and presented at the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2021)[2].

The content from the aforementioned publication is used under the terms of Creative Commons Attribution 4.0 International License.[3]

Statement of Personal Contribution

The author of this thesis substantially contributed to all aspects of the previous publication (Namysl, Behnke, and Köhler, 2021), including the conception, design, and implementation of the proposed methods, the preparation of the data for training and evaluation of the proposed approach, conducting the experimental evaluation, the analysis and interpretation of the experimental results, drafting the manuscript, as well as the revision and final approval of the version to be published.

The content presented in this chapter, unless otherwise stated, is the contribution of the author of this thesis.

Unpublished Content

Compared to the previous publication, additional, unpublished content is included in this chapter. In particular, the analysis of the data efficiency of the proposed

[1] ©2021 Association for Computational Linguistics. Reprinted in Appendix C.3.
[2] `https://2021.aclweb.org`
[3] `https://creativecommons.org/licenses/by/4.0`

method is additionally presented in Section 4.6.5.

Abstract

Despite recent advances, standard sequence labeling systems often fail when processing noisy user-generated text or consuming the output of an OCR process. In this chapter, the confusion matrix-based noise model used in the original NAT method is replaced by an empirical error generation approach that employs a sequence-to-sequence model trained to perform translation from error-free to erroneous text. Using an OCR engine, a large parallel text corpus is generated for training and several real-world noisy sequence labeling benchmarks are produced for evaluation. Moreover, to overcome the data sparsity problem that exacerbates in the case of imperfect textual input, noisy language model-based embeddings are additionally integrated into the NAT framework. The presented approach outperformed the baseline noise generation and error correction techniques on the erroneous sequence labeling data sets.

4.1 Introduction

Deep learning models have already surpassed human-level performance in many NLP tasks (Devlin et al., 2019; Z. Zhang et al., 2019).[4] Sequence labeling systems have also reached extremely high accuracy (Akbik, Bergmann, and Vollgraf, 2019; Heinzerling and Strube, 2019). Still, NLP models often fail in scenarios, where nonstandard text is given as input (Belinkov and Bisk, 2018; Heigold et al., 2018).

NLP systems are predominantly trained on error-free input but are also employed to process user-generated text (Baldwin et al., 2013; Derczynski et al., 2013) or consume the output of prior OCR and ASR processes (Miller et al., 2000). Both misrecognized and mistyped text pose a challenge for the standard models trained using error-free data (see Section 3.6.1). It is worth noting that the errors occurring in any upstream component of an NLP system deteriorate the accuracy of the target downstream task, diminishing the performance of the whole IE system (Alex and Burns, 2014).

This chapter is primarily focused on the problem of performing sequence labeling on the text produced by an OCR engine. Moreover, the transferability of the methods learned to model OCR noise to the distribution of the human-generated

[4] GLUE benchmark (A. Wang et al., 2018): `https://gluebenchmark.com/leaderboard`

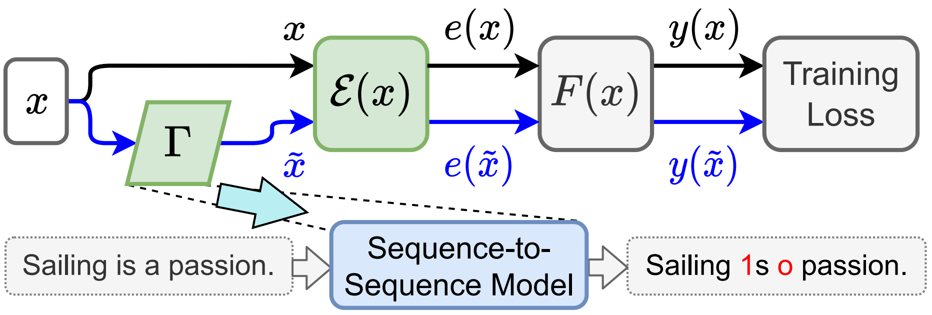


Figure 4.1: The proposed modification of the NAT method (green boxes). Firstly, the confusion matrix-based noise model is replaced with a learnable sequence-to-sequence error generator. Secondly, the FLAIR embeddings (Akbik et al., 2018) are retrained using noisy text to improve the accuracy of noisy neural sequence labeling. Γ is a process that induces noise to the input x producing erroneous $\tilde{x}$. $\mathcal{E}(x)$ is an embedding matrix. $F(x)$ is a sequence labeling system. $e(x)$ and $e(\tilde{x})$ are the embeddings of x and $\tilde{x}$, respectively. $y(x)$ and $y(\tilde{x})$ are the outputs of the system for x and $\tilde{x}$, respectively.

errors is also investigated. Different from the method described in Chapter 3, this chapter presents an improved empirical error generation approach that imitates real-world OCR error patterns during the training of a NAT model more precisely. The underlying assumption is that fidelity to real-world noise distribution should enhance the robustness of the noisy neural sequence labeling process beyond the improvements achieved by the baseline NAT approach.

In conclusion, this chapter makes the following contributions (Figure 4.1):

- Firstly, a targeted noise generation method for OCR is proposed. It employs a sequence-to-sequence model trained to translate from error-free to erroneous text (Section 4.4.1). The presented approach improves the accuracy of noisy neural sequence labeling compared to the standard NAT method (Section 4.6.1).
- Secondly, an unsupervised parallel training data generation method, which directly utilizes an OCR engine, is presented (Section 4.4.2). Similarly, realistic noisy versions of popular sequence labeling data sets can be synthesized for evaluation (Section 4.5.6), mitigating the *data scarcity problem*.
- Thirdly, to overcome the *data sparsity problem*, the erroneous text is used to perform noisy language modeling (NLM) (Section 4.4.5). The proposed NLM embeddings further improve the accuracy of noisy neural sequence labeling, also in the case of human-generated errors (Sections 4.6.3 and 4.6.4).

4.2 Related Work

In this section, a brief review of related approaches in the field of text correction is presented. Please refer to Section 3.6 for a thorough overview of other methods for improving robustness to nonstandard input.

4.2.1 Spelling and OCR Postcorrection

The most widely used method of handling noisy text is to apply error correction on the input produced by human writers (*spelling correction*) or the output of an upstream OCR component (*OCR postcorrection*).

Noisy channel modeling is one of the most widely studied correction techniques (Brill and Moore, 2000; Duan and Hsu, 2011; Kemighan et al., 1990; Kolak and Resnik, 2002, 2005; Toutanova and Moore, 2002). A popular alternative is to detect the erroneous tokens, generate plausible correction candidates, and rank them in order of correctness (Fivez et al., 2017; Flor et al., 2019).

Another extensively used approach is to adapt machine translation-based methods to the text correction task (Afli et al., 2016; Schmaltz et al., 2017). For instance, Schnober et al. (2016) apply *monotone* sequence-to-sequence modeling to this task. Moreover, Hämäläinen and Hengchen (2019) proposed *Natas* — an OCR postcorrection technique that uses a character-level NMT method. They extracted parallel training data using embeddings learned from the erroneous text and used it as input to their translation model. Other related postcorrection techniques include voting algorithms that align results from multiple OCR engines (Al Azawi et al., 2015; Lund et al., 2011; Wemhoener et al., 2013) and hybrid systems (Schulz and Kuhn, 2017).

This thesis proposes a different approach that attempts to make the neural models robust without relying on prior error correction, which, in the case of OCR errors, is still far from being solved (Chiron et al., 2017; Rigaud et al., 2019).

4.2.2 Grammatical Error Correction

Grammatical error correction (GEC) (Bryant et al., 2019; Ng et al., 2014, 2013) aims to automatically correct ungrammatical text. This task is not limited to typographical errors but also includes cases that involve longer dependencies such as errors in subject-verb agreement. GEC can be approached as a translation from an ungrammatical to a grammatical language, which enabled NMT sequence-to-sequence models to be applied to this task (Yuan and Briscoe, 2016). However, due to the limited size of human-annotated GEC corpora, NMT models could not be trained effectively (Lichtarge et al., 2019), though.

Artificial Error Generation

Several studies investigated generating realistic erroneous sentences from grammatically correct text to boost training data (Choe et al., 2019; Grundkiewicz et al., 2019; Kasewa et al., 2018; Qiu and Park, 2019). Inspired by *backtranslation* (Edunov et al., 2018; Sennrich et al., 2016), artificial error generation (AEG) approaches (Rei et al., 2017; Z. Xie et al., 2018) train an intermediate model in reverse order — to translate correct sentences into the erroneous ones.

Following AEG approaches, a large corpus of clean and noisy sentences is generated for the training of the sequence-to-sequence models employed in this work. As a result, the proposed method produces rich and diverse errors resembling the natural noise distribution (see Sections 4.3.3 and 4.4.2).

4.3 Problem Definition and Motivation

4.3.1 Noisy Neural Sequence Labeling

As pointed out in Sections 3.1 and 4.1, the standard NLP systems are generally trained using error-free textual input, which causes a discrepancy between the training and the test conditions. These systems are thus more susceptible to nonstandard, corrupted, or adversarial input.

To model this phenomenon, in Section 3.2.2, the *noisy neural sequence labeling* problem was formulated. It assumes that every input sentence might be subjected to some unknown token-level noising process $\Gamma = P(\tilde{x}_i|x_i)$, where x_i is the original i-th token and $\tilde{x}_i$ is its distorted equivalent. As a solution, the NAT framework was proposed (Section 3.3). It trains the sequence labeling model using auxiliary objectives that exploit both the original sentences and their copies corrupted using a noising process that imitates the naturally occurring errors.

4.3.2 Confusion Matrix-Based Error Model

In Section 3.3.1, a confusion matrix-based method was used to model insertions, deletions, and substitutions of characters. Given a corpus of paired noisy and manually corrected sentences $\mathcal{P}$, the natural error distribution was estimated by calculating the alignments between the pairs $(\tilde{x}, x) \in \mathcal{P}$ of noisy and clean sentences using the *Levenshtein distance* metric (Levenshtein, 1966).

Moreover, as $\mathcal{P}$ is usually laborious to obtain, a *vanilla error model* was also defined and employed to induce noise into error-free sentences during the training of the NAT-based models. This model assumes that all types of edit operations are equally likely (see Equation (3.1)).

4.3.3 Realistic Empirical Error Modeling

In Section 3.5.3, the NAT models that use the vanilla and the empirically estimated confusion matrix-based error models were compared and no advantage of exploiting the test-time error distribution during training was observed. *But would we make the same observation given a more realistic error model?*

Even though the methods that use randomized error patterns were often successful (Section 3.6.2), leveraging the empirical noise distribution for training should be beneficial, providing additional accuracy improvements. The data produced by the naïve noise generation methods may not resemble naturally occurring errors, which could lead the downstream models to learn misleading patterns.

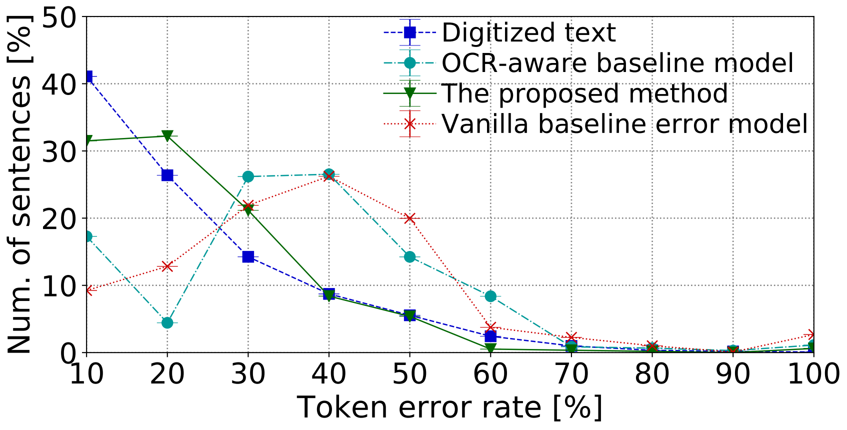


Figure 4.2: Distributions of the token error rates of sentences produced by the baseline confusion matrix-based error models and the method proposed in this chapter. For comparison, the distribution of error rates in the text that contains naturally occurring errors is also included. Each value n on the x-axis is the percentage of sentences with a token error rate in $[n - 10, n)$.

Figure 4.2 compares the distributions of error rates of sentences produced by the method proposed in this chapter with the confusion matrix-based noise models introduced in Section 3.3.1. For comparison, the distribution of error rates in the noisy text that was used to estimate both empirical error models is also included. It can be seen that the distribution of naturally occurring errors follows Zipf's law, while the baseline confusion matrix-based noise models produce bell-shaped curves. Interestingly, both the vanilla and the empirical baselines exhibit similar characteristics, which could explain the observations that were made based on the results of the experiment in Section 3.5.3.

In practice, the error rate is not uniform throughout the text. Some passages are recognized perfectly, while others can barely be deciphered. The objective of this chapter is thus to develop a noise model that produces a smoother distribution, imitating the errors encountered at test time more precisely (see *The proposed method* in Figure 4.2).

Moreover, although the exact noise distribution in the test data cannot always be known beforehand, the noising process, e.g., an OCR engine, used to provide the input for the downstream task, can often be identified. One could thus take advantage of such prior knowledge to improve the efficiency of the complete IE system that is employed to process noisy input data.

4.3.4 Data Sparsity of Natural Language

Embeddings are real-valued vectors that are typically used to represent text passages (words, sentences, paragraphs) in many recent NLP systems (Mikolov, Sutskever, et al., 2013; Pennington et al., 2014). They capture syntactic and semantic textual features that can be exploited to solve higher-level language tasks. Embeddings that are pretrained on a large corpus of mono- or multilingual text became ubiquitous (Akbik et al., 2018; Devlin et al., 2019; Peters et al., 2018).

Embeddings are generally trained using corpora that contain error-free text. Due to the data sparsity problem that arises from the large vocabulary sizes and the exponential number of feasible contexts, the majority of possible word sequences do not appear in the input data. Even though increasing the size of the training corpora was shown to improve the performance of language processing tasks (Brown et al., 2020), most of the misrecognized or mistyped tokens would still be unobserved and therefore poorly modeled when using the error-free text only. Therefore, in this chapter, we also seek the answer to the following question: *Would it be beneficial to pretrain the embeddings on data that includes realistic erroneous sentences?*

4.3.5 The Issues of Error Correction Methods

Furthermore, we can assume that the correction methods, although widely adopted, can only reliably manage moderately perturbed text (Flor et al., 2019). In particular, OCR postcorrection has been reported to be very challenging in the case of historical books that exhibit high OCR error rates (Chiron et al., 2017; Rigaud et al., 2019).

Note that the correction methods have no information about the downstream task to be performed. Moreover, in the automatic correction setting, they only provide the best guess for each token. Comparing their performance with the NAT approach in the context of sequence labeling would be thus informative.

4.4 Proposed Empirical Error Modeling Method

Figure 4.1 summarizes the modifications of the NAT framework presented in this chapter. Specifically, the following improvements were introduced:

- Firstly, the confusion matrix-based noising process was replaced with a noise induction method that generates a more realistic error distribution (Sections 4.4.1 to 4.4.4).
- Secondly, to overcome the data sparsity problem discussed in Section 4.3.4, the language model-based embeddings were trained using digitized text (Section 4.4.5) and used as a substitution for the pretrained model proposed in Akbik et al. (2018).

4.4.1 Sequence-to-Sequence Error Generator

Motivated by the AEG approaches (Rei et al., 2017; Z. Xie et al., 2018), a learnable error generation method is proposed in this section. This method employs a character-level, sequence-to-sequence model to perform monotone string translation (Schnober et al., 2016). It directly models the conditional probability $p(\tilde{x}|x)$ of mapping error-free text x into erroneous text $\tilde{x}$ using an attention-based encoder-decoder framework (Bahdanau et al., 2015). The encoder computes the representation $h = \{h_1, \ldots, h_n\}$ of x, where n is the length of x. The decoder generates $\tilde{x}$ one token at a time:

$$p(\tilde{x}|x) = \prod\nolimits_{i=1}^{n} p(\tilde{x}_i|\tilde{x}_{<i}, x, c), \tag{4.1}$$

$$c = f_{attn}(\{h_1, \ldots, h_n\}), \tag{4.2}$$

where c is a vector generated from h, and f_{attn} is an attention function. The employed sequence-to-sequence models are trained to maximize the likelihood of the training data. At inference time, the subsequent tokens are sampled from the learned conditional language model.

Moreover, note that the presented approach reverses the standard sequence-to-sequence error correction pipeline, which uses the erroneous text as input and trains the model to produce the corresponding error-free string (Figure 4.3). By interchanging the input and the output data, sentence correction models can also be readily trained. One difference is that, at inference time, it would be preferred to perform a beam search and select the best decoding result rather than sampling subsequent characters from the learned distribution.

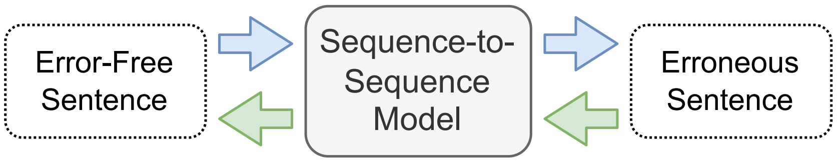


Figure 4.3: Schematic visualization of the error generation (blue arrows) and the error correction (green arrows) methods. The parallel data can be utilized to train sequence-to-sequence models for both tasks.

4.4.2 Unsupervised Parallel Data Generation

To train the error generation model introduced in Section 4.4.1, a large parallel corpus $\mathcal{P}$ of error-free and erroneous sentences is required. AEG approaches use seed GEC corpora to learn the inverse models directly. Unfortunately, a comparably large resource for the digitized text that could be used for this task, can hardly be found.

To address this issue, an unsupervised sentence-level parallel data generation approach for OCR is proposed (Figure 4.4). The proposed method assumes that a large seed corpus $\mathcal{T}$ that contains the error-free text and a set of fonts $\mathcal{F}$ have been collected. Each sentence from the corpus $\mathcal{T}$ is first rendered as an image and then subsequently recognized using an OCR engine. Moreover, to increase the variation in training data, different fonts are sampled from $\mathcal{F}$ and used for rendering. Furthermore, to simulate an imperfect image acquisition process and the degradation of the printed material, both geometrical distortions and pixel-level noise are induced to the images before recognition. Table 4.1 presents examples of a sentence rendered using different noising settings. When induced simultaneously, geometrical distortions and pixel-level noise pose the greatest challenge to the subsequent OCR process.

Note that the presented approach is universal and could be used to generate parallel data sets for other tasks, e.g., an ASR system could be trained on samples from a text-to-speech engine (D. Wang et al., 2018).

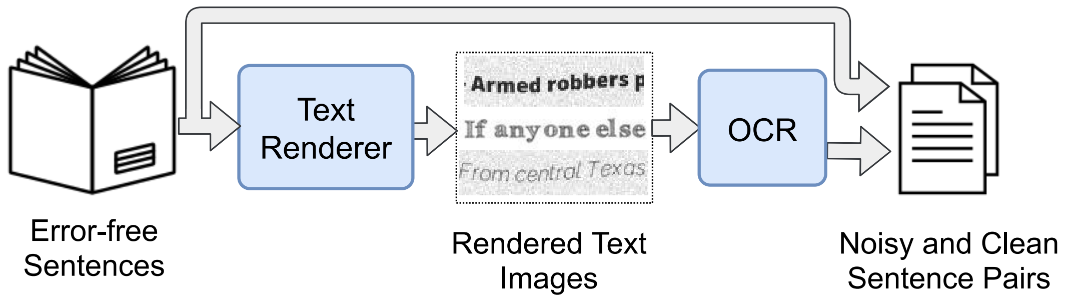
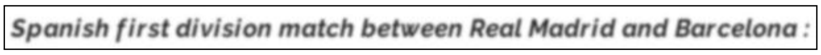
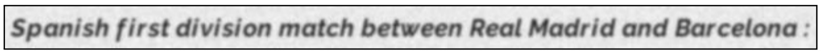
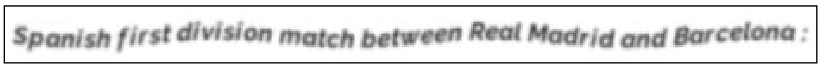
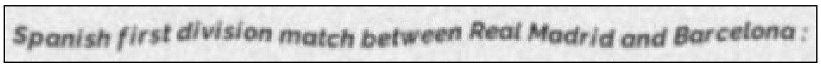


Figure 4.4: The proposed parallel data generation method for OCR. The sentences extracted from a text corpus are rendered as images. Subsequently, an OCR engine recognizes the text depicted in the rendered images. Finally, the pairs of original and recognized sentences are gathered to form a parallel corpus used to train translation models for error generation and correction.

Table 4.1: Examples of a sentence rendered using different noising settings employed in the parallel data generation method (Section 4.4.2). Geometrical distortions (the 3rd and 4th row) and pixel-level noise (the 2nd and 4th row) were induced to the original image (the 1st row).

Rendered Sentence	Geom. Distort.	Pixel-Level Noise
 Spanish first division match between Real Madrid and Barcelona :	✗	✗
 Spanish first division match between Real Madrid and Barcelona :	✗	✓
 Spanish first division match between Real Madrid and Barcelona :	✓	✗
 Spanish first division match between Real Madrid and Barcelona :	✓	✓

4.4.3 Sentence- and Token-Level Error Modeling

Note that the sequence labeling problem is formulated at the token level, i.e., each token has a class label assigned to it (Section 3.2.1). To employ the method proposed in Section 4.4.1 in this scenario: (1) the *sentence-level* and (2) the *token-level* variant of the error generator are further developed.

Sentence-Level Error Generator

The sentence-level error generator uses a sequence-to-sequence model trained to translate from error-free to erroneous sentences. It can potentially utilize contextual information from surrounding tokens, which may improve the quality of the results. During the training of a NAT model, a learned sequence-to-sequence model translates the original input x to generate $\tilde{x}$. Subsequently, an alignment algorithm introduced in Section 4.4.4 is used to transfer the token-level annotations from x to $\tilde{x}$.

Token-Level Error Generator

The token-level error generator uses a sequence-to-sequence model trained to translate from error-free to erroneous tokens. It relies exclusively on the input and the output tokens and discards the context, which could potentially provide relevant information for the generator, especially in the case of short words, e.g., prepositions or conjunctions. The alignment algorithm is used to build a training set for this task, i.e., extract token-level parallel data from the corpus of parallel sentences (Section 4.4.2). During the training of a NAT model, a learned generator translates each token x_i from x to produce the erroneous sentence $\tilde{x}$.

4.4.4 Token-Level Sentence Alignment

Figure 4.5 illustrates the alignment procedure that was developed to extract token-level parallel training data for the token-level generator and to transfer the labels to the erroneous sentences for the sentence-level generator in the sequence labeling scenario. To this end, each pair of error-free and noisy sentences is aligned at the token level using the Levenshtein distance algorithm (Levenshtein, 1966). The proposed alignment procedure takes both the error-free and the erroneous sentences as input and produces pairs of aligned tokens. The annotations for tokens are transferred accordingly.

4.4.5 Noisy Language Modeling

Recently, Z. Xie et al. (2017) drew a connection between input noising in neural network language models and smoothing in n-gram models. Data noising could be

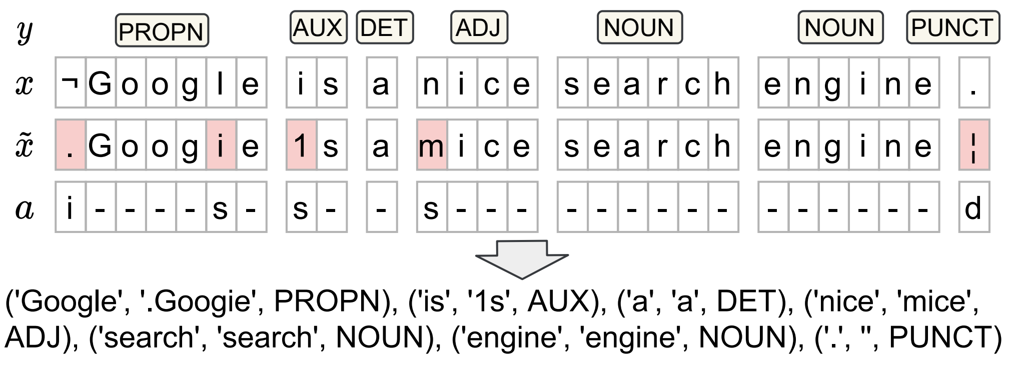


Figure 4.5: The proposed sentence alignment procedure. The original and the recognized sentences (x and $\tilde{x}$, respectively) are aligned using the sequence of edit operations a, which includes insertions i, deletions d, and substitutions s of characters. The symbols '¬' and '¦' are used as placeholders for the insertion and the deletion operation, respectively. Matched characters are marked with '−'. The alignment procedure produces a list of paired error-free and possibly erroneous tokens with class labels y (optional). The original sentence used in this example was extracted from the UD English EWT data set (Table B.5). Tokens are annotated with the corresponding part of speech.

an effective technique for regularizing neural language models that would help to overcome the data sparsity problem of imperfect natural language text and enable learning meaningful representation of erroneous tokens.

To this end, the proposed method includes the data from noisy sources in the corpora used to train language model-based embeddings. Specifically, a noisy language model is learned using the output of an OCR engine (Section 4.4.2) that captures the characteristics of OCR errors. Any other noisy source could be readily used to model related domains, e.g., ASR transcripts or ungrammatical text.

The NLM embeddings are used as a replacement for the FLAIR embeddings introduced by Akbik et al. (2018). Similarly, transformer-based language models could also be employed. Compared to the work of Piktus et al. (2019), language model-based embeddings assign each token a representation based on its context and are thereby better suited to the task of modeling noisy sentences.

4.5 Experimental Setup

In this section, the employed experimental setup is thoroughly described to facilitate reproducibility of the experimental evaluation presented in Section 4.6.

4.5.1 Sequence-to-Sequence Error Generation and Correction

Hyperparameters

To learn the error generation and error correction models (Section 4.4.1), the OpenNMT toolkit[5] (Klein et al., 2017) is utilized in this work. Table A.4 in the appendix lists all nondefault hyperparameters that were employed to train the sequence-to-sequence models used in the experiments. Moreover, it is worth noting that the learning rate was decayed eight times during the training for all sequence-to-sequence models. Furthermore, the *copy attention* (See et al., 2017) and the *global attention* (Luong et al., 2015) functions were employed as f_{attn} used in Equation (4.2) in the case of the error generation and the error correction models, respectively.

Sentence Encoding-Decoding Schema

The OpenNMT toolkit was primarily designed to process data tokenized at the word level. To adapt it to the character-level input, a sentence encoding-decoding process depicted in Figure 4.6 was employed. Specifically, the input sentence is first encoded at the character level before feeding it into the sequence-to-sequence model. Subsequently, the output produced by the sequence-to-sequence model is decoded back to the original form.

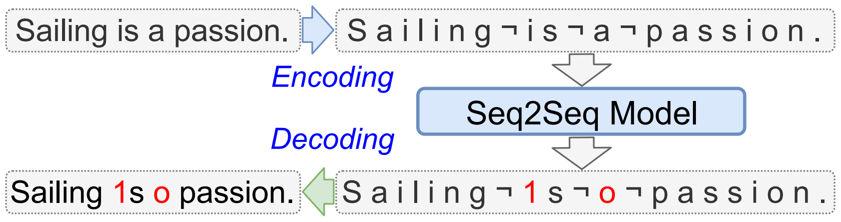

Sailing is a passion.
S a i l i n g ¬ i s ¬ a ¬ p a s s i o n .
Encoding
Seq2Seq Model
Decoding
Sailing 1s o passion.
S a i l i n g ¬ 1 s ¬ o ¬ p a s s i o n .

Figure 4.6: Sentence encoding-decoding schema. The whitespace characters are first replaced with a placeholder symbol '¬'. The sentences are tokenized at the character level by adding whitespace between every pair of characters. Decoding reverses this process.

Validation Accuracy and Learnable Parameters

Table 4.2 summarizes the validation accuracy of the sequence-to-sequence models for error generation. The sentence-level models were trained for 1.6×10^4 and the token-level models for 4×10^5 iterations or at least one epoch of training.

[5] https://github.com/OpenNMT/OpenNMT-py

Moreover, the token-level error correction model employed by Natas was trained for one epoch (about 4×10^5 iterations) on one million parallel sentences and achieved 96.9% accuracy on the validation set of 5,000 sentences. Furthermore, all sequence-to-sequence models for error generation and correction employed in the experiments have about 7.7 million parameters.

Table 4.2: Validation accuracy of the sequence-to-sequence models for error generation. Both the token-level and the sentence-level variants were trained. The first and the second column shows the number of parallel sentences used for training and validation, respectively.

Training Set Size	Validation Set Size	Validation Accuracy	
		Token-Level	Sentence-Level
10^7	5,000	98.3%	95.7%
10^6	5,000	95.4%	94.9%
10^5	5,000	95.1%	95.3%
10^4	1,000	94.6%	90.1%
10^3	100	93.3%	91.6%

4.5.2 Unsupervised Parallel Data Generation

Following the approach introduced in Section 4.4.2, a large parallel corpus $\mathcal{P}$ was generated to train the error generation and correction models. Firstly, ten million sentences, which accounts for about 253 million words, were sampled from the English part of the *1 Billion Word Language Model Benchmark*[6] and used as the source of error-free text, i.e., the seed corpus $\mathcal{T}$. Each sentence from $\mathcal{T}$ was rendered as an image using the *Text Recognition Data Generator* package[7]. Moreover, 90 different fonts were collected as $\mathcal{F}$ and used for rendering. Furthermore, random distortions were induced to the rendered images. Subsequently, OCR was performed on each image of text using a Python wrapper[8] for Tesseract OCR[9] (Smith, 2007).

Finally, the pairs of error-free and erroneous sentences were gathered to form the parallel corpus $\mathcal{P}$. Note that, to be consistent with the tokenization rules applied in the FLAIR framework (Akbik, Bergmann, Blythe, et al., 2019), *segtok*[10]

[6] `https://github.com/ciprian-chelba/1-billion-word-language-modeling-benchmark`
[7] `https://pypi.org/project/trdg`
[8] `https://github.com/sirfz/tesserocr`
[9] Tesseract v4.0 was used to generate the parallel data set $\mathcal{P}$.
[10] `https://github.com/fnl/segtok`

was used to tokenize the error-free sentences prior to the alignment procedure (Section 4.4.4).

The generated parallel data corpus $\mathcal{P}$ was used to train both the sentence-level and the token-level error generation models (Section 4.4.3). For error correction, only the context-free models were trained, as the applied OCR postcorrection method could only be performed at the token level.

4.5.3 Noisy Language Modeling

FLAIR (Akbik et al., 2018) learns a bidirectional language model to represent sequences of characters. To train the NLM embeddings proposed in this work (Section 4.4.5), the target side of the parallel corpus $\mathcal{P}$ was employed to retrain FLAIR embeddings on the noisy digitized text. The hyperparameters used to train the language model were consistent with prior work of Akbik et al. (2018).[11]

4.5.4 Sequence Labeling

Training Setup

The NAT framework[12] (Figure 4.1) was employed to study the robustness of sequence labeling systems. Following Akbik et al. (2018), a combination of *FLAIR* and *GloVe* embeddings was used in all experiments.[13] The data augmentation and the stability training objectives ($\mathcal{L}_{\text{augm}}$ and $\mathcal{L}_{\text{stabil}}$, respectively) were employed with default weights ($\alpha = 1.0$). Consistent with the experimental setup used in Section 3.4, erroneous sentences $\tilde{x}$ were generated dynamically in every epoch.

Evaluation Setup

The evaluation pipeline is shown in Figure 4.7. Following Akbik et al. (2018), the entity-level micro-average F_1 score and the per token accuracy is used as the evaluation metrics for NER and POST, respectively.

Learnable Parameters

The number of parameters of the sequence labeling models is constant across different variants, as the same architecture is used in all experiments. The number

[11] Note that they also used the 1 Billion Word corpus for pretraining.
[12] `https://github.com/mnamysl/nat-acl2020`
[13] Other hyperparameters also follow Akbik et al. (2018).

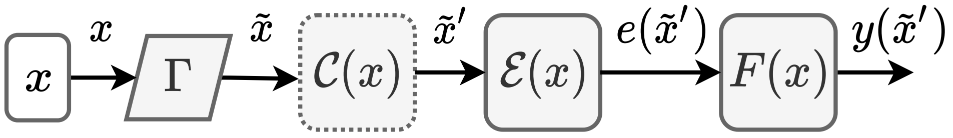


Figure 4.7: Evaluation pipeline. Γ is a noising process that transforms x into $\tilde{x}$. $\mathcal{C}(x)$ is an optional text correction module that returns $\tilde{x}'$ ($\tilde{x}' = \tilde{x}$, if $\mathcal{C}(x)$ is absent). $\mathcal{E}(x)$ is an embedding matrix. $F(x)$ is a sequence labeling system. $e(\tilde{x}')$ and $y(\tilde{x}')$ are the embeddings and the output of the system for $\tilde{x}'$, respectively.

of all parameters is 60.3 million (including embeddings that were fixed during the training), and the number of all trainable parameters is 25.5 million.

Computing Architecture and Average Runtime

The training and the evaluation were performed on a workstation equipped with an Intel Xeon CPU with ten cores and an Nvidia Quadro RTX 6000 graphics card with 24 gigabytes of memory. The evaluation of the complete test set took seven and ten seconds on average in the case of the UD English EWT and the English CoNLL 2003 data set, respectively. The runtime did not depend on the training method that was used. Nevertheless, when the correction method was employed, the runtime was significantly lengthened, e.g., it took almost three minutes to evaluate a model that employed the Natas correction method on the English CoNLL 2003 data set.

4.5.5 Tasks and Data Sets

The NER and POST tasks were used to examine the sequence labeling models. NER aims to locate all named entities mentioned in the input text and classify them into predefined classes, e.g., person names, locations, and organizations. POST is the process of tagging each token in the text with the corresponding part of speech.

For NER, the English CoNLL 2003 data set (Tjong Kim Sang and De Meulder, 2003)[14] was employed. To evaluate POST, the Universal Dependency Treebank (UD English EWT; Silveira et al., 2014)[15] was used. The detailed statistics of the aforementioned data sets are presented in Tables B.1a and B.5 in the appendix.

[14] https://www.clips.uantwerpen.be/conll2003/ner
[15] https://universaldependencies.org/treebanks/en_ewt (version 2.6)

4.5.6 Noisy Benchmarks

Data Scarcity Problem

Unfortunately, there is no publicly available noisy sequence labeling data set that could be used as a benchmark to compare the accuracy of different methods for improving robustness. Such a data set should be preferably generated using a realistic noising process to precisely imitate the real-world conditions. Many previous approaches employed synthetic noise patterns for evaluation (Heigold et al., 2018) or randomly sampled errors from the lookup tables of possible lexical replacements (Belinkov and Bisk, 2018). However, these methods cannot accurately reflect the real-world input to be encountered at test time (Section 4.3.3).

OCR Noise

To perform a realistic evaluation, several noisy versions of the original sequence labeling data sets were generated. The sentences were extracted from each original benchmark and, subsequently, the procedure described in Section 4.4.2 was applied.[16] Furthermore, the token-level annotations were transferred as described in Section 4.4.4. Finally, as a result, the data in the CoNLL format was produced. Table 4.3 shows an excerpt from a noisy sequence labeling data set generated for evaluation.

Table 4.3: An example of a sentence from the noisy English CoNLL 2003 data set. The first and the second column contain the noisy and the error-free tokens, respectively. The third column denotes the class label in BIO format, where the **B**eginning-, **I**nside-, and **O**utside-of-entity subtags are distinguished.

Noisy Token	Error-Free Token	Class Label
`No`	`No`	`O`
`nzw`	`new`	`O`
`fixtuvzs`	`fixtures`	`O`
`reported`	`reported`	`O`
`from`	`from`	`O`
`New`	`New`	`B-LOC`
`Vork`	`York`	`I-LOC`
`.`	`.`	`O`

[16] Both Tesseract v3.04 and v4.0 were applied. Note that different sets of distortions and image backgrounds were used than those employed to generate parallel training data.

Human-Generated Errors

Moreover, to evaluate the transferability of error generators, the experimental setup employed in Section 3.4.4 was used, i.e., the misspellings were synthetically induced into the error-free data sets. To this end, the lookup tables of possible lexical replacements released by Belinkov and Bisk (2018) and Piktus et al. (2019) were used.[17]

Summary of Noisy Benchmarks

As a result, each noisy data set exhibits a different level of difficulty for the methods examined in the experiments. In summary, the following noisy versions of both original data sets were additionally generated for evaluation (Table 4.4):

- **Tesseract 3$^\clubsuit$** - generated using a less accurate Tesseract 3.04 OCR engine on undistorted images with a clean background.
- **Tesseract 4$^\diamondsuit$** - generated using the Tesseract 4.0 OCR engine on distorted images with a noisy background.
- **Tesseract 4$^\heartsuit$** - generated using the Tesseract 4.0 OCR engine on distorted images with a clean background.
- **Tesseract 4$^\spadesuit$** - generated using the Tesseract 4.0 OCR engine on undistorted images with a noisy background.
- **Typos** - generated by applying an out-of-domain distribution of human-generated errors.

Table 4.4: The noisy sequence labeling data sets that were generated either by applying OCR on rendered sentences from an original benchmark (first four rows) or by inducing misspellings (last row). Multiple variants of the former data sets were generated by combining geometrical distortions and pixel-level noise induction. The last two columns present the token error rates (the column headers indicate the name of the original benchmark).

Data Set	Geometrical Distortions	Pixel-Level Noise	English CoNLL 2003	UD English EWT
Tesseract 3$^\clubsuit$	✗	✗	22.72%	23.31%
Tesseract 4$^\diamondsuit$	✓	✓	16.35%	22.12%
Tesseract 4$^\heartsuit$	✓	✗	14.89%	20.38%
Tesseract 4$^\spadesuit$	✗	✓	3.53%	5.83%
Typos	n/a	n/a	15.53%	15.22%

[17] Both sets were merged and used to induce typos into the original data (see Section 4.6.4).

4.5.7 Error Generation Baselines

The proposed token-level and sentence-level error generators were compared with the OCR-aware confusion matrix-based noise model described in Section 4.3.2 (marked as *Baseline* in Tables 4.5, 4.6, 4.9 and 4.10). For a fair comparison, the noisy part of the parallel corpus $\mathcal{P}$ was used to estimate the confusion matrix employed by this baseline. Moreover, in the experiments that involved the proposed NLM embeddings, the vanilla error model described in Section 3.3.1 (marked as *Vanilla* in Tables 4.9 and 4.10b) was also used.

4.5.8 Error Correction Baselines

In this scenario, the sequence labeling models were trained using the standard objective $\mathcal{L}_0$. Subsequently, a text correction method was applied to the erroneous input before feeding it into the sequence labeling model (see Figure 4.7). Two error correction methods were examined: *Hunspell*[18], a widely adopted spell checker, and *Natas*[19] (Hämäläinen and Hengchen, 2019), a sequence-to-sequence OCR postcorrection method. The latter model was trained to correct OCR errors in 18th-century documents, thus it could not be used directly. Therefore, the context-free error correction models compatible with Natas were retrained using the parallel corpus $\mathcal{P}$ (Section 4.5.1). Following the authors, the default ONMT hyperparameters were employed, except that a BRNN was used as the encoder.

4.6 Experimental Results

4.6.1 Empirical Noise Generation Approaches

In this experiment, the NAT models that employed either the proposed sequence-to-sequence error generators (Section 4.4.3) or the baseline confusion matrix-based noise model (Section 4.3.2) were compared. In this evaluation scenario, the $\mathcal{C}(x)$ module is omitted (Figure 4.7).

Tables 4.5 and 4.6 present the results of this experiment. The proposed error generators consistently outperformed the OCR-aware confusion matrix-based model on the noisy benchmarks generated using the Tesseract 4.0 engine. Note that Tesseract 4.0 was used to produce the parallel corpus $\mathcal{P}$, which was employed to estimate the confusion matrix used by the baseline noise model and to train

[18] `https://hunspell.github.io`
[19] `https://github.com/mikahama/natas`

the sequence-to-sequence error generators. On the other hand, the advantage of the proposed method over the baseline approach was less emphasized in the case of the noisy data sets generated using the Tesseract 3.04 engine.

Table 4.5: Comparison of error generation approaches on the original and noisy English CoNLL 2003 test sets (Section 4.6.1). Four noisy variants of the original benchmark were used: Tesseract 3$^{\clubsuit}$, Tesseract 4$^{\diamondsuit}$, Tesseract 4$^{\heartsuit}$, Tesseract 4$^{\spadesuit}$ (Table 4.4). Mean and standard deviation F_1 scores over five runs with different random initialization are reported. The *Noise Model* column corresponds to the noise generator used at training time: the *Baseline* OCR-aware confusion matrix-based noise model (Section 4.3.2) or the *Sentence-Level** and *Token-Level** sequence-to-sequence error generators (Section 4.4.3). **Bold** values indicate top results (within the models trained using the same objective) that are statistically inseparable (Welch's t-test; $p < 0.05$).

(a) Data Augmentation Training Objective ($\mathcal{L}_{augm}$)

Noise Model	Original Data	Tesseract 3$^{\clubsuit}$	Tesseract 4$^{\diamondsuit}$	Tesseract 4$^{\heartsuit}$	Tesseract 4$^{\spadesuit}$
Baseline	92.56±0.06	**85.29±0.16**	88.62±0.08	89.19±0.12	92.04±0.07
Token-Level*	**92.76±0.07**	**85.38±0.16**	**89.39±0.17**	**89.99±0.22**	**92.37±0.10**
Sentence-Level*	**92.81±0.11**	84.38±0.15	88.96±0.18	**89.67±0.26**	**92.44±0.17**

(b) Stability Training Objective ($\mathcal{L}_{stabil}$)

Noise Model	Original Data	Tesseract 3$^{\clubsuit}$	Tesseract 4$^{\diamondsuit}$	Tesseract 4$^{\heartsuit}$	Tesseract 4$^{\spadesuit}$
Baseline	92.23±0.12	**84.49±0.10**	87.58±0.13	88.40±0.20	91.65±0.14
Token-Level*	**92.24±0.18**	**84.25±0.23**	**88.24±0.25**	**88.91±0.21**	**91.86±0.16**
Sentence-Level*	**92.45±0.12**	83.89±0.30	**88.14±0.23**	**88.88±0.11**	**91.99±0.11**

Comparing the results obtained by both variants of the proposed sequence-to-sequence error generator (Section 4.4.3), the token-level translation method performed better than the sentence-level variant, while the latter was more efficient when the error rate of the input was lower (see the *original data* and the *Tesseract 4*$^{\spadesuit}$ columns), although it often struggled with translating long sentences. Moreover, data augmentation generally outperformed stability training, which is consistent with the results obtained in Section 3.5.1.

Furthermore, we observe a slight decrease in accuracy on the original UD English EWT data set with both auxiliary objectives. This could be caused by the different proportions of the tokens that were perturbed during training by the proposed sequence-to-sequence error generators (e.g., 18% and 19.5% in the case of the token-level model for the English CoNLL 2003 and the UD English EWT data set, respectively). The trade-off between accuracy for clean and noisy data has

Table 4.6: Comparison of error generation approaches on the original and noisy UD English EWT test sets (Section 4.6.1). Four noisy variants of the original benchmark were used: Tesseract 3♣, Tesseract 4♢, Tesseract 4♡, Tesseract 4♠ (Table 4.4). Mean and standard deviation accuracies over five runs with different random initialization are reported. The *Noise Model* column corresponds to the noise generator used at training time: the *Baseline* OCR-aware confusion matrix-based noise model (Section 4.3.2) or the *Sentence-Level** and *Token-Level** sequence-to-sequence error generators (Section 4.4.3). **Bold** values indicate top results (within the models trained using the same objective) that are statistically inseparable (Welch's t-test; $p < 0.05$).

(a) Data Augmentation Training Objective ($\mathcal{L}_{augm}$)

Noise Model	Original Data	Tesseract 3♣	Tesseract 4♢	Tesseract 4♡	Tesseract 4♠
Baseline	**96.90±0.06**	**91.35±0.13**	92.12±0.14	92.99±0.21	96.17±0.07
Token-Level*	96.76±0.04	**91.44±0.11**	**93.65±0.13**	**94.19±0.10**	**96.26±0.07**
Sentence-Level*	96.78±0.06	90.92±0.08	93.37±0.08	**94.10±0.03**	**96.27±0.03**

(b) Stability Training Objective ($\mathcal{L}_{stabil}$)

Noise Model	Original Data	Tesseract 3♣	Tesseract 4♢	Tesseract 4♡	Tesseract 4♠
Baseline	**96.80±0.04**	91.16±0.07	91.93±0.11	92.77±0.10	96.06±0.02
Token-Level*	96.65±0.07	**91.36±0.12**	**93.34±0.09**	**93.97±0.05**	**96.14±0.07**
Sentence-Level*	96.67±0.05	90.70±0.14	93.05±0.17	93.71±0.13	**96.15±0.05**

thus been shifted toward the latter. We can also notice a greater advantage of the sequence-to-sequence error generation method over the baseline on the noisy UD English EWT data sets than in the case of the English CoNLL 2003 benchmark.

4.6.2 Error Generation vs. Error Correction

This experiment compares the NAT approach with the baseline error correction methods applied to the input text prior to feeding it into the sequence labeling model (see $\mathcal{C}(x)$ in Figure 4.7). Table 4.7 presents the results of this experiment.

Preliminary analysis revealed that these baselines underperformed on both the original and the noisy data sets due to the *overcorrection* problem. To make them more competitive, their default dictionaries were extended by adding all tokens from the corresponding test sets for evaluation. Although the vocabulary of a test set could rarely be entirely determined, this setting would simulate a scenario where accurate, in-domain vocabularies could be exploited by the correction methods.

Table 4.7: Comparison of error correction approaches on the original and noisy English CoNLL 2003 and UD English EWT test sets (Section 4.6.2). Four noisy variants for each original data set were used: Tesseract 3$^{\clubsuit}$, Tesseract 4$^{\diamondsuit}$, Tesseract 4$^{\heartsuit}$, Tesseract 4$^{\spadesuit}$ (Table 4.4). Mean and standard deviation F_1 scores (English CoNLL 2003) and accuracies (UD English EWT) over five runs with different random initialization are reported. All models were trained using the standard objective $\mathcal{L}_0$. The *Correction Method* column corresponds to the correction approach executed before the target task prediction: *Hunspell* or *Natas*. **Bold** values indicate top results that are statistically inseparable (Welch's t-test; $p < 0.05$).

(a) English CoNLL 2003 Test Set

Correction Method	Original Data	Tesseract 3$^{\clubsuit}$	Tesseract 4$^{\diamondsuit}$	Tesseract 4$^{\heartsuit}$	Tesseract 4$^{\spadesuit}$
—	**92.54±0.08**	80.48±0.09	84.71±0.19	85.62±0.08	91.50±0.08
Hunspell	**92.54±0.08**	**82.17±0.11**	**85.80±0.11**	**86.70±0.07**	**91.73±0.11**
Natas	**92.54±0.08**	77.80±0.19	84.50±0.11	85.24±0.10	91.33±0.13

(b) UD English EWT Test Set

Correction Method	Original Data	Tesseract 3$^{\clubsuit}$	Tesseract 4$^{\diamondsuit}$	Tesseract 4$^{\heartsuit}$	Tesseract 4$^{\spadesuit}$
—	**96.96±0.04**	86.75±0.16	86.97±0.14	88.30±0.16	94.34±0.07
Hunspell	**96.96±0.04**	87.53±0.14	86.74±0.14	88.12±0.16	94.49±0.08
Natas	**96.96±0.04**	**88.98±0.10**	**88.94±0.14**	**89.68±0.16**	**95.11±0.08**

A direct comparison of the results obtained by the error correction methods (Table 4.7) with the scores of the error generation methods presented in Tables 4.5 and 4.6 reveals that, although more general, error correction techniques were outperformed by the NAT approach regardless of the noising method used. This can be attributed to the fact that the correction methods have no information about the downstream task to be performed. In contrast, the NAT method is directly integrated into the training process of the downstream task, which provides more information to be exploited to solve this task more efficiently and results in higher overall accuracy.

The comparison between the examined correction methods showed that, surprisingly, Hunspell performed better than Natas on the English CoNLL 2003 data set. To better understand this finding, a thorough inspection of the results of both methods was carried out. Table 4.8 presents the error rates and the correction accuracies of the Natas and Hunspell methods calculated on the test sets of both noisy sequence labeling benchmarks.

Table 4.8: Token Error Rates (TER) and the correction accuracies (ACC) of Natas and Hunspell on the test sets of the noisy sequence labeling data sets. In the case of the English CoNLL 2003 data set, the correction accuracies computed exclusively on the entity tokens, marked as ACC (entities), are additionally presented. All values are percentages. **Bold** values represent the lowest token error rates and the highest accuracies.

(a) English CoNLL 2003

Metric	Method	Tesseract 3$^\clubsuit$	Tesseract 4$^\diamondsuit$	Tesseract 4$^\heartsuit$	Tesseract 4$^\spadesuit$	Typos
TER	Original	22.72	16.35	14.89	3.53	15.53
	Natas	**17.24**	**12.20**	**11.13**	**2.34**	11.53
	Hunspell	17.44	13.54	12.24	2.43	**10.69**
TER (entities)	Original	29.66	16.70	15.00	3.61	8.20
	Natas	27.81	14.97	13.36	2.93	7.62
	Hunspell	**16.63**	**9.95**	**8.76**	**1.89**	**4.07**
ACC	Natas	**24.13**	**25.40**	**25.24**	**33.70**	25.75
	Hunspell	23.26	17.19	17.76	31.20	**31.17**
ACC (entities)	Natas	6.23	10.41	10.93	18.77	7.07
	Hunspell	**43.93**	**40.44**	**41.58**	**47.78**	**50.38**

(b) UD English EWT

Metric	Method	Tesseract 3$^\clubsuit$	Tesseract 4$^\diamondsuit$	Tesseract 4$^\heartsuit$	Tesseract 4$^\spadesuit$	Typos
TER	Original	23.31	22.12	20.38	5.83	15.22
	Natas	**17.76**	**17.46**	**16.23**	**4.21**	11.68
	Hunspell	19.14	19.74	18.09	4.75	**11.22**
ACC	Natas	**23.82**	**21.05**	**20.36**	**27.75**	23.27
	Hunspell	17.90	10.74	11.20	18.59	**26.49**

Although Natas achieved higher overall accuracy, it struggled with the correction of tokens that were a part of named entities. This behavior could be an issue of data-driven error correction methods, as the entities are relatively rare in written text and are often out-of-vocabulary tokens (Alex and Burns, 2014).

To further verify the assumption about the cause of the inferior accuracy of the downstream NER task in the case when the Natas method was used as preprocessing, a set of unique entity tokens was additionally extracted from the test set of the English CoNLL 2003 data set. As a result, 20% of these tokens were not covered and 31% appeared at most three times in the data that was employed to train the error correction model, which confirms the assumption formulated in the previous paragraph.

4.6.3 Noisy Language Modeling

This experiment compares the accuracy of the NAT models with the vanilla noise generator (Section 4.3.2) that employed either the pretrained FLAIR embeddings or the proposed NLM embeddings (Sections 4.4.5 and 4.5.3). In this evaluation scenario, the $\mathcal{C}(x)$ module is omitted (Figure 4.7).

Note that the noise model and the embeddings are two distinct components of the NAT architecture (Γ and $\mathcal{E}(x)$ in Figure 4.1, respectively) and therefore they could be easily used in combination with each other. However, in this work, the NLM embeddings are not mixed with empirically estimated error models to avoid the twofold empirical error modeling effect and the evaluation of this combination is left to future work.

Table 4.9 summarizes the results of this experiment. The proposed NLM embeddings significantly improved the accuracy across all training objectives, even when the standard training objective for the sequence labeling task ($\mathcal{L}_0$) was employed. Surprisingly, the evident improvements were also achieved for the noisy data set generated using the Tesseract 3.04 engine, which confirms that the NLM embeddings can effectively model the features of erroneous tokens from an out-of-domain noise distribution.

On the other hand, the NLM embeddings slightly decreased the accuracy on the original data for the standard training objective compared to the baseline pretrained embeddings. This effect could be investigated in future work by eliminating possible differences in the pretraining procedure and comparing the NLM against a model trained on the original, error-free text corpus instead of using the pretrained embeddings released by Akbik et al. (2018).

Table 4.9: Comparison of the NAT models trained with and without the proposed NLM embeddings (Section 4.6.3) on the original and noisy English CoNLL 2003 test sets. Four noisy variants of the original data set were used: Tesseract 3♣, Tesseract 4♢, Tesseract 4♡, Tesseract 4♠ (Table 4.4). Mean and standard deviation F_1 scores over five runs with different random initialization are reported. The *Noise Model* column corresponds to the *Baseline* OCR-aware or *Vanilla* confusion matrix-based error models (Section 3.3.1). The *NLM* column indicates whether the model employed the NLM embeddings. **Bold** values indicate top results (within the models trained using the same objective) that are statistically inseparable (Welch's t-test; $p < 0.05$).

(a) Standard Training Objective ($\mathcal{L}_0$)

Noise Model	NLM	Original Data	Tesseract 3♣	Tesseract 4♢	Tesseract 4♡	Tesseract 4♠
n/a	✗	**92.54±0.08**	80.48±0.09	84.71±0.19	85.62±0.08	91.50±0.08
n/a	✓	92.09±0.07	**83.83±0.21**	**88.17±0.12**	**88.71±0.17**	**91.68±0.09**

(b) Data Augmentation Training Objective ($\mathcal{L}_{augm}$)

Noise Model	NLM	Original Data	Tesseract 3♣	Tesseract 4♢	Tesseract 4♡	Tesseract 4♠
Baseline	✗	**92.56±0.06**	85.29±0.16	88.62±0.08	89.19±0.12	92.04±0.07
Vanilla	✗	92.39±0.11	85.59±0.23	88.01±0.17	88.65±0.20	91.93±0.13
Vanilla	✓	92.45±0.05	**87.28±0.19**	**90.12±0.19**	**90.43±0.19**	**92.17±0.05**

(c) Stability Training Objective ($\mathcal{L}_{stabil}$)

Noise Model	NLM	Original Data	Tesseract 3♣	Tesseract 4♢	Tesseract 4♡	Tesseract 4♠
Baseline	✗	**92.23±0.12**	84.49±0.10	87.58±0.13	88.40±0.20	**91.65±0.14**
Vanilla	✗	92.04±0.06	84.63±0.17	87.24±0.24	88.02±0.10	**91.52±0.12**
Vanilla	✓	91.85±0.07	**86.79±0.11**	**89.32±0.12**	**89.77±0.05**	**91.51±0.07**

4.6.4 Human-Generated Errors

This experiment evaluates the utility of the proposed sequence-to-sequence error generators learned to model OCR noise and the proposed NLM embeddings in a scenario where the input contains human-generated errors. For evaluation, the noisy data sets with synthetically induced misspellings were used (Section 4.5.6). Moreover, the $\mathcal{C}(x)$ module is omitted in this scenario (Figure 4.7).

Table 4.10 summarizes the results of this experiment. The models that employ the proposed NLM embeddings outperformed the baselines for all training objectives. Moreover, the sequence-to-sequence error generation approach performed on par with the OCR-aware confusion matrix-based models on the English CoNLL 2003 data set, while the latter achieved better accuracy on the UD English EWT data set. Nevertheless, the sequence-to-sequence error generation method also proved to be beneficial in this scenario, which would suggest that the errors made by human writers and by the text recognition engines have common characteristics that were successfully exploited by this method.

The observed difference in accuracy of the sequence-to-sequence error generator could be caused by the discrepancy between the data distributions. Note that although the data used in this experiment reflects the patterns of human-generated errors, the distribution of these errors does not necessarily follow the natural distribution of human-generated errors, as it was synthetically generated using a fixed replacement probability that was uniform across all candidates.

Figure 4.8 presents the distribution of token error rates in relation to the number of sentences in the noisy data sets. Note that the error distribution of the noisy data sets generated using the unsupervised parallel data generation method proposed in Section 4.4.2 is close to the Zipfian distribution. As the *Typos* data set was generated by randomly sampling possible lexical replacement candidates from the lookup tables, its distribution exhibits a bell-shaped curve pattern. This observation suggests that the data sets generated using the proposed method are better suited for the evaluation of the robustness of sequence labeling models than the data generated by the prior, lookup table-based approaches.

4.6.5 Relationship with the Size of the Parallel Corpus

Empirical error generators are especially beneficial when we can approximate the noise distribution to be encountered at test time. This experiment aims to answer the question of how much parallel training data is required to train a solid sequence-to-sequence error generation model.

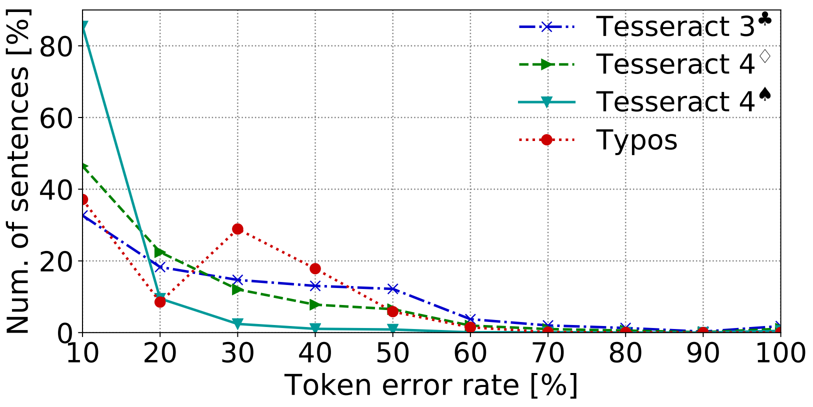
Table 4.10: Transferability of the error generators and the embeddings learned to model OCR noise to the distribution of the human-generated errors (Section 4.6.4). The evaluation was performed on the English CoNLL 2003 and the UD English EWT test sets with synthetically induced typos. Mean and standard deviation F_1 scores (English CoNLL 2003) and accuracies (UD English EWT) over five runs with different random initialization are reported. $\mathcal{L}_0$, $\mathcal{L}_{augm}$, $\mathcal{L}_{stabil}$ is the standard, the data augmentation, and the stability objective, respectively. The *Noise Model* column corresponds to the error model employed at training time: either the *Sentence-Level** and *Token-Level** sequence-to-sequence error generators, or the *Baseline* OCR-aware and *Vanilla* confusion matrix-based noise model (Section 3.3.1). The *NLM* column indicates whether the model employed the NLM embeddings. **Bold** values indicate top results (within the models trained using the same objective) that are statistically inseparable (Welch's t-test; $p < 0.05$).

(a) Empirical Error Generation Methods

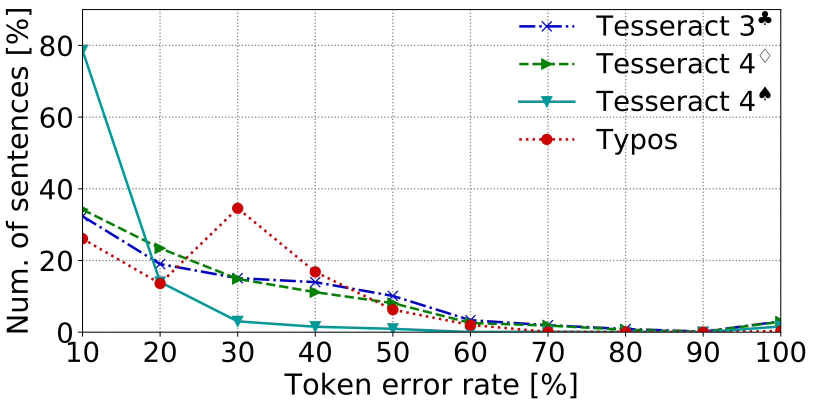
Training Loss	Noise Model	English CoNLL 2003 with Typos	UD English EWT with Typos
$\mathcal{L}_0$	n/a	88.79±0.07	90.54±0.11
$\mathcal{L}_{augm}$	Baseline	**90.82±0.12**	**93.63±0.11**
	Token-Level*	**90.92±0.13**	92.87±0.08
	Sentence-Level*	**90.77±0.19**	92.68±0.09
$\mathcal{L}_{stabil}$	Baseline	**90.30±0.13**	**93.37±0.05**
	Token-Level*	**90.19±0.12**	92.79±0.08
	Sentence-Level*	**90.15±0.16**	92.42±0.11

(b) The Impact of the NLM Embeddings

Training Loss	Noise Model	NLM	English CoNLL 2003 with Typos
$\mathcal{L}_0$	n/a	✗	88.79±0.07
	n/a	✓	**89.60±0.24**
$\mathcal{L}_{augm}$	Baseline	✗	90.82±0.12
	Vanilla	✗	90.77±0.14
	Vanilla	✓	**91.10±0.05**
$\mathcal{L}_{stabil}$	Baseline	✗	90.30±0.13
	Vanilla	✗	90.34±0.06
	Vanilla	✓	**90.53±0.07**



(a) Noisy sentence labeling data sets (English CoNLL 2003)



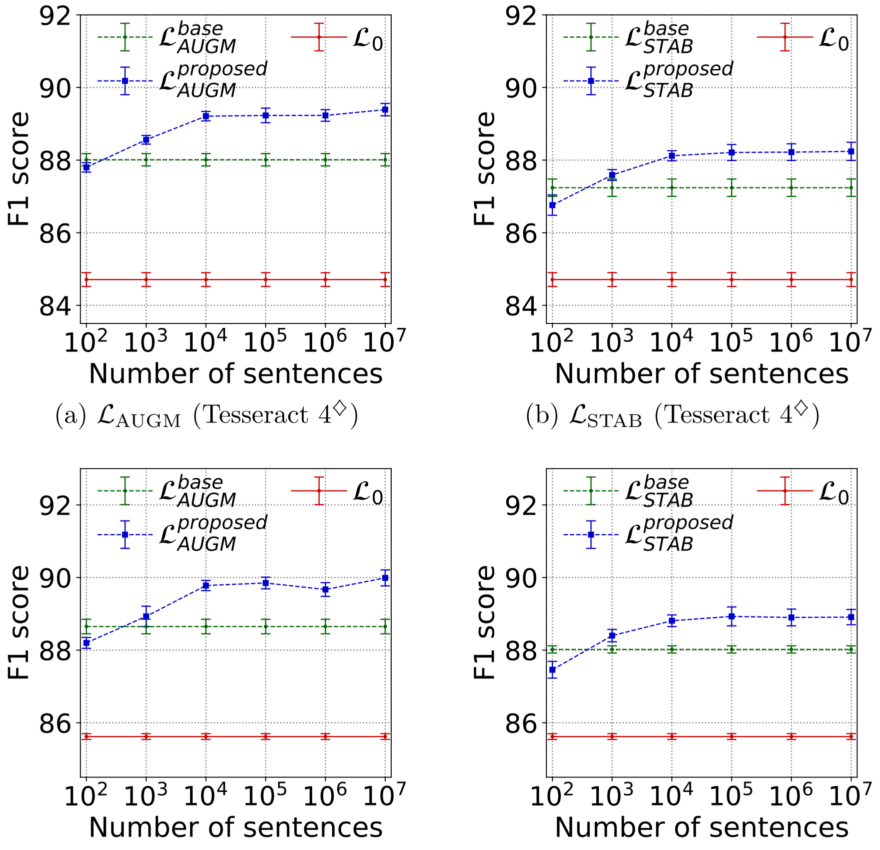
(b) Noisy sentence labeling data sets (UD English EWT)

Figure 4.8: Distributions of the token error rates of sentences in the noisy sequence labeling data sets: Tesseract 3♣, Tesseract 4♢, Tesseract 4♠, and Typos (Section 4.5.6). Each value n on the x-axis is the percentage of sentences with a token error rate in $[n-10, n)$, e.g., the value of 50 corresponds to the sentences with an error rate greater than 40 and lower than or equal to 50.

To this end, multiple NAT models that employ either the sequence-to-sequence error generator or the vanilla confusion matrix-based noise model (Section 3.3.1) were trained. The latter noise model does not need any training data and was used as a noise-aware baseline. The original English CoNLL 2003 data set and its noisy variants with OCR errors and human-generated typos were used for evaluation. The amount of training data was incrementally increased to find the level above which the empirical error models start to consistently outperform the baseline.

Figures 4.9 and 4.10 present the results of this experiment. The NAT models that employed the proposed sequence-to-sequence error generation approach performed better than the models that used the baseline vanilla error model for all noisy benchmarks that were generated using the Tesseract 4.0 OCR engine (Figure 4.9). The improvements were observed when as few as 1,000 parallel training sentences were used. Unexpectedly, the proposed method also outperformed the baseline on the original data set (Figure 4.10).

On the contrary, the results in Figure 4.10 show that the accuracy of the NAT models trained using the proposed error generator fell slightly behind the variants that employed the baseline noise model on the noisy data sets generated using the Tesseract 3.04 OCR engine (*Tesseract 3*♣) and the data with human-generated errors (*Typos*).



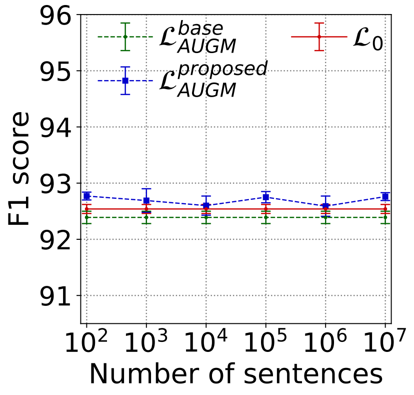
(a) $\mathcal{L}_{\text{AUGM}}$ (Tesseract 4♢)

(b) $\mathcal{L}_{\text{STAB}}$ (Tesseract 4♢)

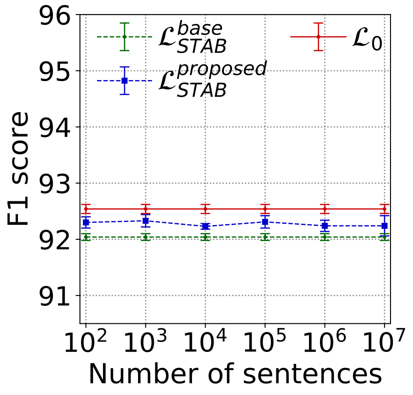
(c) $\mathcal{L}_{\text{AUGM}}$ (Tesseract 4♡)

(d) $\mathcal{L}_{\text{STAB}}$ (Tesseract 4♡)

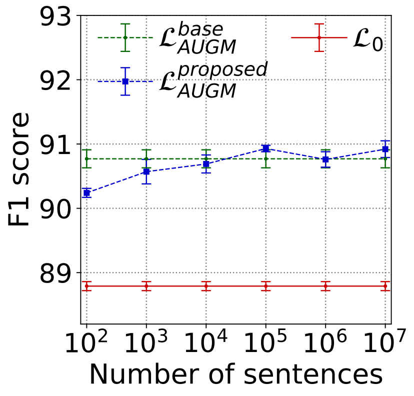
Figure 4.9: F_1 score in relation to the number of parallel sentences (Section 4.6.5). Two noisy variants of the English CoNLL 2003 data set were used: Tesseract 4♢ and Tesseract 4♡. The token-level sequence-to-sequence method is compared with the vanilla error model, and the standard objective $\mathcal{L}_0$. The results for the data augmentation ($\mathcal{L}^{\text{proposed}}_{\text{AUGM}}$, $\mathcal{L}^{\text{base}}_{\text{AUGM}}$) and the stability training ($\mathcal{L}^{\text{proposed}}_{\text{STAB}}$, $\mathcal{L}^{\text{base}}_{\text{STAB}}$) objectives are presented.



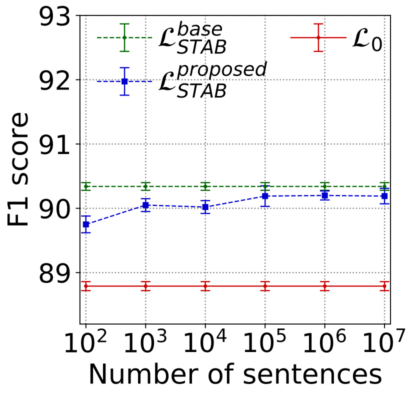
(a) $\mathcal{L}_{\text{AUGM}}$ (original data set)



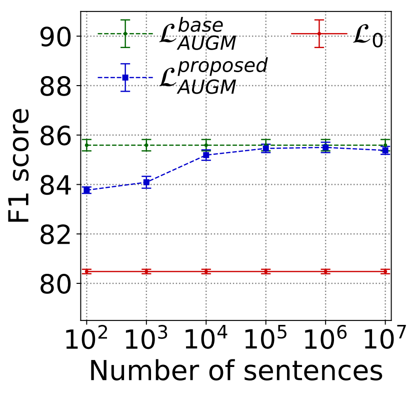
(b) $\mathcal{L}_{\text{STAB}}$ (original data set)



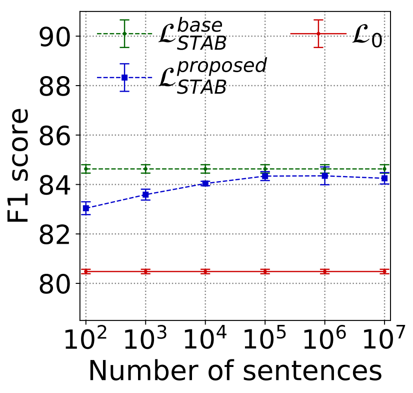
(c) $\mathcal{L}_{\text{AUGM}}$ (Typos)



(d) $\mathcal{L}_{\text{STAB}}$ (Typos)



(e) $\mathcal{L}_{\text{AUGM}}$ (Tesseract 3♣)



(f) $\mathcal{L}_{\text{STAB}}$ (Tesseract 3♣)

Figure 4.10: F_1 score in relation to the number of parallel sentences (Section 4.6.5). The original English CoNLL 2003 benchmark and its noisy variants: Tesseract 3♣, and Typos were used. The token-level sequence-to-sequence approach is compared with the vanilla error model, and the standard objective $\mathcal{L}_0$. The results for the data augmentation ($\mathcal{L}_{\text{AUGM}}^{\text{proposed}}$, $\mathcal{L}_{\text{AUGM}}^{\text{base}}$) and the stability training ($\mathcal{L}_{\text{STAB}}^{\text{proposed}}$, $\mathcal{L}_{\text{STAB}}^{\text{base}}$) objectives are presented.

4.7 Summary

In this chapter, the task of performing sequence labeling on noisy digitized and human-generated text was thoroughly studied. The NAT approach introduced in Section 3.3 was extended by integrating the empirical error generator that performs the translation from error-free to erroneous text (Section 4.4.1). To train the proposed generator, the unsupervised parallel data synthesis method that directly employs an OCR engine was developed (Section 4.4.2). Analogously, several realistic noisy benchmarks were produced for evaluation (Section 4.5.6). Moreover, the NLM embeddings were introduced in Section 4.4.5. These embeddings substantially alleviate the data sparsity problem of natural language that exacerbates in the case of imperfect textual input.

The presented approach outperformed the baseline noise induction and error correction methods, improving the accuracy of the noisy neural sequence labeling task (Section 4.6). In particular, it was demonstrated that the representation learned to model OCR noise is transferable to the out-of-domain scenarios — the human-generated error distribution (Section 4.6.4) and the noise induced by a different OCR engine (Sections 4.6.1 and 4.6.3). Finally, the proposed error generator was shown to be data-efficient (Section 4.6.5), which facilitates its practical application.

Future Work Directions

Grundkiewicz and Junczys-Dowmunt (2019) showed that unsupervised systems benefit from domain adaptation on authentic labeled data. Therefore, future work could involve fine-tuning the NAT models that were pretrained on synthetic samples using the labeled data generated directly by the natural noising process.

Moreover, the incorporation of the NAT approach into every framework that processes noisy input text is strongly advocated. The NAT approach should also improve the performance of other NLP tasks beyond the sequence labeling scenario. In future work, the NAT method and the NLM embeddings should preferably be evaluated using an NLP benchmark that covers diverse language understanding tasks. Furthermore, the research community would further benefit from pretrained multilingual NLM embeddings, as well as from a broader spectrum of models specialized in a particular language or error distribution.

5 Flexible Table Recognition and Semantic Interpretation

Preface

This chapter extends the approach introduced in Namysl, Esser, Behnke, and Köhler (2022)[1] that was previously published by SciTePress and was also presented at the 17th International Conference on Computer Vision Theory and Applications (VISAPP 2022)[2].

Statement of Personal Contribution

The author of this thesis substantially contributed to all aspects of the previous publication (Namysl, Esser, Behnke, and Köhler, 2022), including the conception, design, and implementation of the proposed methods, the preparation of the data for training and evaluation of the proposed approach, conducting the experimental evaluation, the analysis and interpretation of the experimental results, drafting the manuscript, as well as the revision and final approval of the version to be published.

The content presented in this chapter, unless otherwise stated, is the contribution of the author of this thesis.

Abstract

Table extraction is an important but still unsolved problem. This chapter presents a flexible and modular table extraction system that supports the most frequent table layouts and handles both the documents in image format and PDF files with embedded text. The presented approach combines a deep learning-based table detection module, heuristics for table structure recognition (TSR), and a

[1] ©2022 SciTePress. Reprinted in Appendix C.4.
[2] https://visapp.scitevents.org/?y=2022

rule-based semantic interpretation method. The proposed system achieves results competitive with state-of-the-art approaches on two challenging benchmarks from the table competitions held at the International Conference on Document Analysis and Recognition (ICDAR). Noteworthy, an issue in the evaluation script used in a recent competition was corrected and the results of the proposed and the baseline method were reported using both the original and the rectified script. Finally, the effectiveness of the presented table extraction system was demonstrated in the scenario, where raw documents were given as input and the target information was contained in a subset of table columns.

5.1 Introduction

Information can hardly be presented in a more compressed way than in a table. Humans can easily comprehend documents containing tables (Wright, 1980). In contrast, automatic table extraction has not been completely solved yet, although it has been widely studied before (see Section 5.3). Due to the heterogeneity of document formats (e.g., invoices, scientific papers, or balance sheets), this task is extremely hard. However, as the number of digitized documents steadily increases, a solution for automatic IE from tabular data is urgently needed.

In this chapter, a holistic approach is introduced. It combines table recognition and semantic interpretation modules, which allows performing IE directly from tables in documents that are either in image format or contain embedded text. Specifically, in Section 5.4, a basic variant of the proposed table extraction system is introduced. It combines a heuristic-based table recognition approach, i.e., table detection and structure recognition performed in one step, with a semantic interpretation module implemented as a rule-based method that matches table content with predefined semantic concepts.

For table recognition, two heuristics are proposed that target fully bordered (Section 5.4.3) and partially bordered tables (Section 5.4.4), respectively. The former method handles the most popular table format, which is predominantly used in business documents. The latter algorithm recognizes tables that are typeset with a $\LaTeX$ package that is widely used in scientific and technical publications. By combining these two heuristics, the presented system is able to precisely recognize the two most widely used tabular formats. Moreover, in Section 5.5, the basic variant of the proposed system is extended by incorporating a deep learning-based table detection module and combining it with the adapted version of the heuristics for structure recognition employed in the basic variant of the presented system.

The proposed modifications result in a highly effective, hybrid table recognition approach, which recognizes the structure of borderless and hybrid-layout tables.

The utility of the proposed hybrid table recognition approach is demonstrated on two challenging benchmarks from ICDAR 2013 (Göbel et al., 2013) and ICDAR 2019 (L. Gao et al., 2019) Table Competition that contain documents in different formats (Section 5.7). In both scenarios, the proposed method exhibits recognition accuracy competitive with state-of-the-art approaches. Moreover, in the preliminary experiments, a previously unnoticed issue in the official evaluation tool employed in ICDAR 2019 Table Competition was discovered. The issue in the script was rectified and the corrected version of the code was used for evaluation. The repository with the corrected script was made publicly available and all changes were also submitted to the official evaluation tool[3]. In addition, the evaluation in Section 5.7 also includes the revised annotations used in this competition that were recently released[4]. Furthermore, to facilitate the reproducibility and fair comparison of the results obtained by different methods on the ICDAR 2013 benchmark, the evaluation script employed in this work was also released publicly (Section 5.7.1). It parses the output produced by the official evaluation tool and accumulates them to get the final averaged scores. It also includes the adjacency relations from the false-positively detected tables to give a better perspective on the actual performance of the table recognition approaches.

Concerning the semantic interpretation of the table content, in this work, the basic formulation of the table recognition task is complemented by including a table interpretation module. To this end, a general formulation of the table interpretation task as a *maximum weight matching* (Edmonds, 1965) on a corresponding graph is provided (Section 5.2.3) and a rule-based table interpretation method is proposed (Section 5.4.5). This method leverages regular expressions (RegEx) and an approximate string-matching algorithm to compute semantic similarities between table cells and predefined semantic concepts. By combining the table recognition and table interpretation modules, the presented table extraction system can directly extract the desired information from tables contained in raw, unstructured documents (Section 5.8).

The approach presented in this chapter is both: (1) *flexible*, supporting both documents in image format and PDF files with embedded text, and (2) *modular*, allowing us to separately adapt particular modules to a specific scenario. Both are crucial for a table extraction system because: (1) Few table recognition methods

[3] `https://github.com/cndplab-founder/ctdar_measurement_tool/pull/1`
[4] `https://github.com/cndplab-founder/ICDAR2019_cTDaR/commit/66c411710a99b75a6b07f9cabce2a9480af98c78`

support both types of input, while most approaches require PDF documents with embedded text, and (2) Different processing steps need to be optimized, depending on the document type and the layout of the extremely heterogeneous input data. For some documents, the challenge might be table detection, for others TSR, or semantic interpretation.

The contributions presented in this chapter can be summarized as follows:

- A formal definition of the table extraction task and its main components — table detection, table structure recognition, and table interpretation — is provided in Section 5.2.
- The basic variant of the proposed table extraction system is introduced in Section 5.4. For the recognition of fully and partially bordered tables, two heuristic-based methods are implemented, as described in Sections 5.4.3 and 5.4.4, respectively. For semantic interpretation, a rule-based method is proposed (Section 5.4.5).
- Subsequently, the basic variant of the proposed system is extended by integrating a deep learning-based table detection module and adapting the TSR component from the basic variant of the presented system to enable recognition of borderless and hybrid-layout tables (Section 5.5.2).
- A thorough evaluation is performed using two widely adopted table recognition benchmarks demonstrating the utility of the proposed table recognition method that exhibits recognition accuracy on par with the state-of-the-art approaches (Section 5.7).
- An issue in the official evaluation script used in ICDAR 2019 Table Competition is corrected[5] and the scores of the proposed and the baseline method are reported in all scenarios that involve the original and the corrected script as well as the original and the recently revised annotations.
- To facilitate reproducibility and fair comparison of the results obtained by different table recognition methods, the resources from the experiments presented in this chapter and the evaluation script employed in the experiment on the ICDAR 2013 benchmark have been made publicly available.[6]
- Finally, the presented table extraction system was shown to be effective in a scenario, where the unstructured, scientific documents were given as input and the target information was contained in tables (Section 5.8). The evaluation script employed in this scenario was released publicly to foster future research on IE from tables contained in unstructured documents.[7]

[5] https://github.com/mnamysl/ctdar_measurement_tool/tree/table_mapping_fix
[6] https://github.com/mnamysl/tabrec-sncs
[7] https://github.com/mnamysl/table-interpretation

5.2 Table Extraction Task

Table extraction can be considered as a three-step process consisting of table detection, structure recognition, and interpretation.

The goal of the *table detection* task is to locate all table regions within the input document. Subsequently, TSR aims to recognize the structure of each detected table. Note that both tasks can be performed on different input levels: text lines, words, characters, or pixels.[8] Moreover, although table detection and TSR aim to solve different problems, some approaches cover these two tasks jointly. In this case, we refer to the joint table detection and structure recognition as the *table recognition* process.

Finally, the goal of *table interpretation* is to link the recognized cells with their semantic representation. This step strongly depends on the actual use case and no method fits all scenarios. In this work, this problem is formulated as maximum weight matching (Edmonds, 1965) on a graph with the nodes that correspond to table cells and predefined semantic concepts.

In the following, the table detection, TSR, and table interpretation tasks, that are studied in this work, are described in more detail.

5.2.1 Table Detection

Table detection aims to locate all tables within an input document and can be considered a single-class *object detection* problem. Moreover, it can be split into two subtasks: (1) classify every input element, e.g., every pixel, as being part of a table or not (*image segmentation*), and (2) merge homogeneous input elements into distinct table regions (*region growing and splitting*).

In particular, region growing and splitting approaches make use of a *heterogeneity criterion* that specifies how similar two inputs are (Haralick and Shapiro, 1985). Specifically, keyword-based approaches look for specific words (like *table* or *figure*) and consider all elements within a specific distance to the keyword as being part of the same table region. In contrast, whitespace-based approaches detect large blank areas around the table and consider all enclosed pixels as a homogeneous table region (Shigarov et al., 2018).

Table detection can be performed on different input levels. For instance, on text line level, region growing and splitting becomes, geometrically, a one-dimensional problem. For a text line, one has to decide whether the lines above and below are similar enough or not to form a common table region.

[8] Oro and Ruffolo (2009) speak of so-called *content elements* that form a table.

5.2.2 Table Structure Recognition

During TSR, the structure of a table, i.e., rows, columns, and cells, is recognized. Note that the previous detection step results in a set of content elements belonging to the table object.

Given a set of input elements E belonging to the table object, TSR aims to map these elements to a regular table grid. Formally, we are looking for a mapping:

$$\begin{gathered} \mu : \{0, \ldots, m-1\} \times \{0, \ldots, n-1\} \longrightarrow \mathcal{P}(E) \\ \mu(i,j) = E_{i,j} \subseteq E, \end{gathered} \tag{5.1}$$

which maps each position within an $m \times n$ table to a content element $e \in E$ or a set of multiple elements $E_{i,j} \subseteq E$, with $\mathcal{P}(E)$ denoting the power set of E.

In a simple case, one coordinate (i, j) is mapped to one single element e. Alternatively, elements can be merged at this step — multiple text lines to one text region — so that they collectively form a cell. Thus, μ generally points to a subset $E_{i,j}$ of elements in E.

To allow cells that span more than one row or column, it is valid that two neighboring coordinates $\big((i, j)$ and $(i, j + 1)\big)$ or $\big((i, j)$ and $(i + 1, j)\big)$ both point to the same element. A resulting table cell consists of all neighboring grid points mapped to the same element. Finally, it is also allowed that a grid point is empty and that μ points to an empty set.

5.2.3 Table Interpretation

In the final table interpretation step, the semantic meaning of the table cells is understood. Formally, there exists a set of cells P and a set of meanings M, so that a cell $p \in P$ is mapped to a meaning $m \in M$.

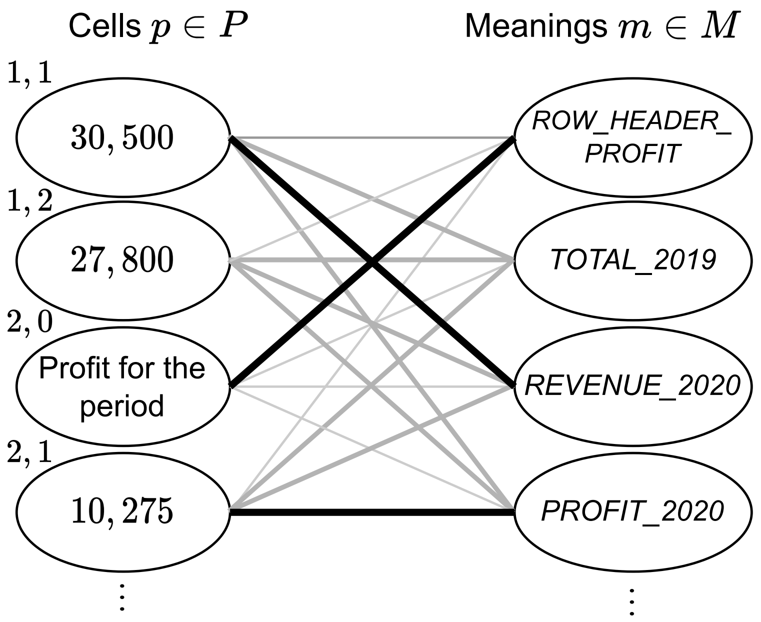
The matching between cells P and meanings M is not necessarily a perfect matching. If a table contains additional columns that are not foreseen in the table model, the cells in these columns cannot be assigned a meaning. On the other hand, when the table model provides optional meanings, some of them cannot always be matched.

For instance, in Figure 5.1, there exists, among others, a meaning *REVENUE_2020*, which specifies the revenue for the fiscal year 2020.[9] During the table interpretation step, one aims to map cell $(1, 1)$ with the content 30,500 to this meaning.

[9] The meanings are denoted in capital letters.

	2020	**2019**
	($'000s)	*($'000s)*
Revenue	**30,500**	27,800
Profit for the period	**10,275**	6,900
Other comprehensive income	**1,125**	1,250
Total comprehensive income	**12,250**	8,633

(a) Balance sheet in tabular form



(b) A table interpretation graph constructed from Figure 5.1a

Figure 5.1: Table interpretation example. **(a)** A financial statement (balance sheet in tabular form). **(b)** The corresponding table interpretation graph. Cells $p \in P$ are mapped to possible meanings $m \in M$. For each mapping, an affinity value is calculated, symbolized by the thickness of the lines.

5.3 Related Work

5.3.1 Complete Table Recognition Approaches

In the following, a review of the most relevant approaches that perform complete table recognition (CTR) is presented. This includes both heuristic-based and learning-based methods performing CTR. For a thorough review of the approaches formerly used for this task, please refer to a comprehensive review presented by Silva et al. (2005).

Heuristic-Based Complete Table Recognition Methods

Heuristic-based approaches implement sets of hand-crafted rules that differ from method to method but generally allow to perform fairly accurate table recognition, given that the format of the tables contained in the input documents is compatible with the designed heuristics. These methods were mainly designed to handle PDF documents with embedded text.

Hassan and Baumgartner (2007) described a system that parses the low-level data from the PDF documents and extracts the HyperText Markup Language (HTML) representation of tables. They locate and segment tables by analyzing the spatial features of text blocks. Their system can detect cells that span multiple rows or columns.

Oro and Ruffolo (2009) introduced *PDF-TREX*, a heuristic, bottom-up method for table recognition in single-column PDF documents. It uses the spatial features of page elements to align and group them into paragraphs and tables. Similarly, it finds the rows and columns and obtains table cells from their intersections.

Nurminen (2013) proposed a set of heuristics for table detection and TSR. Specifically, they locate subsequent text boxes with common left, middle, or right coordinates and assign them the probability of belonging to a table object.

Rastan et al. (2015) presented *TEXUS*, a task-based table processing method. They locate table lines and use transitions between them and main text lines to detect table positions. Moreover, they use alignments of text chunks inside the table region to identify columns and determine the dominant table line pattern to find rows.

More recently, Shigarov et al. (2018) proposed *TabbyPDF*, a heuristic-based approach for table detection and structure recognition from untagged PDF documents. Their system uses both textual and graphical features such as horizontal and vertical distances, fonts, and rulings. Moreover, they exploit the feature of the appearance of text printing instructions and the positions of a drawing cursor.

Learning-Based Complete Table Recognition Methods

Recently, many deep learning-based methods were proposed to solve the image-based table recognition problem. To achieve acceptable results, these approaches need many examples for training. Deep learning methods are often coupled with heuristics that implement the missing functionality.

In particular, Schreiber et al. (2017) proposed *DeepDeSRT* which employs the *Faster R-CNN model* for table detection and a *semantic segmentation* approach for structure recognition. For preprocessing, they stretch the images vertically and horizontally to facilitate the separation of rows and columns by the model. Moreover, they apply postprocessing to fix problems with spurious detections and conjoined regions.

An alternative approach was presented by Reza et al. (2019). They apply conditional *generative adversarial networks* for table localization and an encoder decoder-based model for table row and column segmentation. In their experiments, the segmentation stage was evaluated separately for rows and columns.

Paliwal et al. (2019) proposed *TableNet*, an encoder decoder-based neural architecture for table recognition. Their encoder is shared between the table detection and column segmentation decoders. Subsequently, rule-based row extraction is employed to extract individual cells.

More recently, Prasad et al. (2020) proposed *CascadeTabNet*, which uses the instance segmentation technique to detect tables and segment the cells in a single inference step. Their model predicts the segmentation of cells only for the borderless tables and employs simple rule-based text and line detection heuristics for extracting cells from bordered tables.

Inspired by the approach of Prasad et al. (2020), Fischer et al. (2021) presented a multistage, end-to-end table recognition system named *Multi-Type-TD-TSR* that combines a deep learning-based table detection model with a heuristic-based TSR method. Moreover, they additionally perform a preprocessing step that involves skew angle correction and noise filtering. Furthermore, they perform color normalization prior to the TSR stage to achieve the font and background color invariance.

5.3.2 Table Interpretation Approaches

Table interpretation strongly depends on the actual use case. There is no state-of-the-art method that fits all scenarios, but a variety of approaches from the area of NLP are used. Popular methods involve string matching, calculating the Levenshtein distance (Levenshtein, 1966), or RegEx (Kleene, 1951), e.g., for matching a column title or the data type of a column (Yan and He, 2018). Other methods, like word embeddings (Mikolov, Chen, et al., 2013), entity recognition, relation extraction (Macdonald and Barbosa, 2020), or semantic parsing (Yu et al., 2021), semantically represent table contents. More complex solutions are specifically trained for a certain use case, e.g., a deep learning approach for understanding balance sheets.

Semantic Type Detection

Another task that is related to table interpretation is *semantic type detection*. Semantic types describe the data by providing the correspondence between the columns and the real-world concepts, such as locations, names, organizations, or identification numbers.

A widely adopted method of detecting semantic types is to employ dictionary lookup and RegEx matching of column headers and values. Many popular data preparation and visualization tools successfully incorporate this technique to enhance their data analysis capabilities.[10]

A noteworthy approach was introduced recently by Hulsebos et al. (2019). They proposed *Sherlock* — a deep learning-based method that pairs column headers with 78 semantic types from a knowledge base (Auer et al., 2007). They represent the content of a column using features that describe the distribution of characters, the semantic meaning of the words, and global statistics such as cardinality and uniqueness. In the follow-up work, D. Zhang et al. (2020) proposed a hybrid ML model, which additionally incorporates the column context to predict semantic types. They combine the baseline single-column type prediction used by the Sherlock model with topic modeling and structured prediction to achieve further improvements in recognition accuracy.

5.4 Proposed Basic Table Extraction Method

In this section, the *basic* variant of the proposed table extraction system is described. Figure 5.2 presents the diagram of this system.

We assume that an unstructured document, either an image or a PDF file, is given as input to the system. Firstly, *preprocessing* is performed to prepare the input document for subsequent analysis (Section 5.4.1). Secondly, *table recognition* is executed to detect all tables and recognize their building blocks — rows, columns, and cells. Thirdly, *table interpretation* is carried out to link the extracted structural elements with predefined semantic concepts.

In the presented system, two rule-based table recognition heuristics, that perform table detection and TSR in one step, are implemented. The first method was designed to handle fully bordered tables (Section 5.4.3). It exploits the semi-structured document content that is extracted in the preprocessing step, i.e., the

[10] Popular data analysis systems: `https://powerbi.microsoft.com`, `https://www.trifacta.com`, `https://datastudio.google.com`.

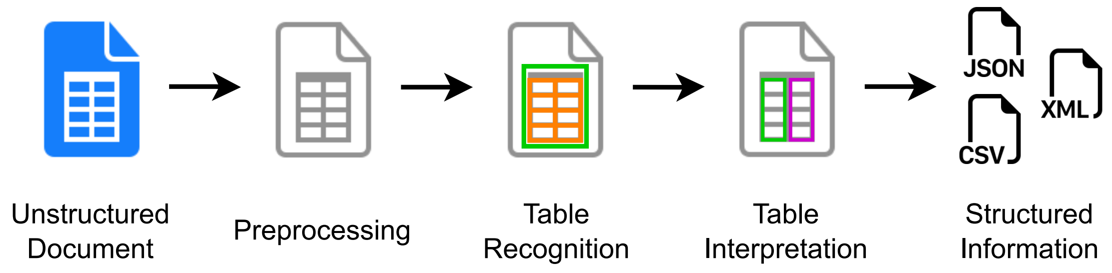


Figure 5.2: The basic variant of the proposed IE system. An unstructured document, either an image or a PDF file, is given as input. Preprocessing is performed prior to table recognition, which detects the table objects and recognizes their building blocks — rows, columns, and individual table cells. Table interpretation links the extracted structural elements with predefined semantic concepts. As a result, the layout and the semantic interpretation of a table is written in structured format.

information about the text blocks and *solid separators*, also called *ruling lines*, to precisely extract the table objects containing cells that are fully outlined. The second algorithm was developed for partially bordered tables (Section 5.4.4). It recognizes tables that are typeset with a commonly used LaTeX package.

Table interpretation (Section 5.4.5) is implemented as a rule-based method that leverages regular expressions and an approximate string matching algorithm to compute similarity between the extracted structural elements of a table and some predefined semantic concepts. Moreover, to link the structural elements with their semantic meanings, a graph-based algorithm is employed.

5.4.1 Preprocessing

Preprocessing is performed to prepare the input document for subsequent analysis. It enables us to work with either PDF files with embedded text or documents in image format. Note that few table recognition methods support both types of input, while most approaches require PDF documents with embedded text (see Section 5.3).

Preprocessing transforms the input document into a semi-structured representation that is exploited by the subsequent components of the presented table extraction system. Specifically, in this work, the layout analysis module described in Konya (2013) is used to extract ruling lines (hereafter referred to as solid separators) and textual page regions from the input document.

In particular, if the input document is in PDF format, it is rendered as an image. The input image is then binarized using a global thresholding method proposed by Otsu (1979). Subsequently, the solid separators are detected on a binary image using a combination of methods described by Y. Zheng et al. (2001) and Gatos et al. (2005). In the case of PDF files with embedded text, the text is directly extracted using a PDF parsing method.[11] Otherwise, OCR is performed using the Tesseract library (Smith, 2007) to extract the textual content from the image.

5.4.2 General Table Recognition Considerations

In this section, general considerations that apply to both table recognition heuristics described in Sections 5.4.3 and 5.4.4 are presented.

Preliminary Steps

As a first step that is common for both proposed table recognition heuristics, an average character size within the input image, denoted as $\overline{S}_x$ and $\overline{S}_y$ for the width and height dimensions, respectively, is calculated using the semi-structured data provided by the preprocessing component. These values are then exploited in the subsequent steps of the proposed heuristics.

Page Orientation

The proposed table recognition heuristics can be easily applied to pages that are oriented horizontally or vertically. The latter case refers to pages that are rotated 90 degrees. In the following sections, the mechanics of each method is described by taking the horizontal layout as the default orientation. Nevertheless, for the vertical layout, all steps are identical, except that the horizontal and the vertical separators are swapped with each other.

The Order of Heuristics

The table recognition heuristics are applied one after the other. The order in which the heuristics are applied impacts the final recognition results because all table candidates that overlap any valid table region that was already detected by the previously applied heuristic are discarded by the employed filtering mechanism. As the heuristic for partially bordered tables could generate spurious candidates from bordered examples, the heuristic for fully bordered tables is applied first followed by the other heuristic.

[11] The presented method employs Poppler (`https://poppler.freedesktop.org`) for both rendering and text extraction.

5.4.3 Recognition of Fully Bordered Tables

Figure 5.3 presents an example of a fully bordered table, which is handled by the rule-based method described in this section.

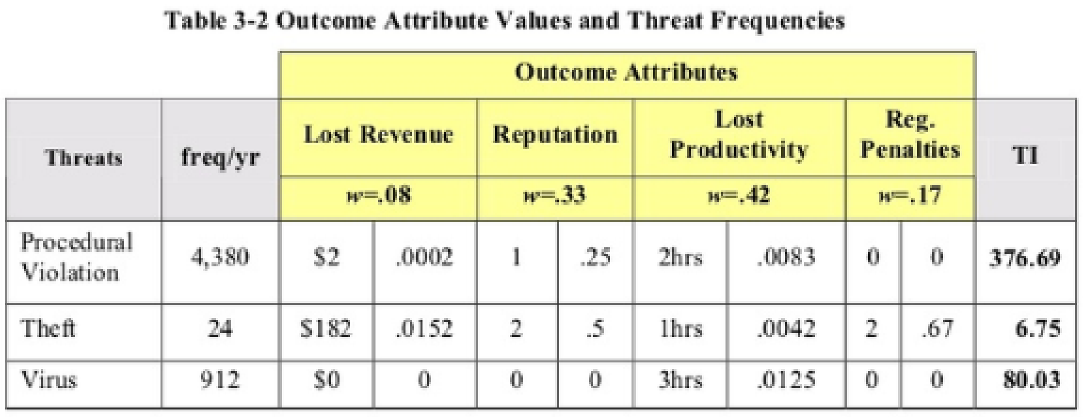


Table 3-2 Outcome Attribute Values and Threat Frequencies

		Outcome Attributes								
Threats	freq/yr	Lost Revenue		Reputation		Lost Productivity		Reg. Penalties		TI
		w=.08		w=.33		w=.42		w=.17		
Procedural Violation	4,380	$2	.0002	1	.25	2hrs	.0083	0	0	**376.69**
Theft	24	$182	.0152	2	.5	1hrs	.0042	2	.67	**6.75**
Virus	912	$0	0	0	0	3hrs	.0125	0	0	**80.03**

Figure 5.3: An example of a fully bordered table cropped from the `cTDaR_t10047.jpg` file contained in the ICDAR 2019 data set (L. Gao et al., 2019).

Separator Merging

The heuristic for fully bordered tables starts by sorting the horizontal and the vertical separators by the top and the left position, respectively. All separator boxes are first expanded by $\delta_x = 5$ and $\delta_y = 5$ pixels to increase the chance of intersection with the neighboring solid separators. Then, all intersecting separators are merged. As a result, they form clusters, as depicted in Figure 5.4.

Finally, all clusters that contain less than one separator with each orientation (vertical and horizontal) are pruned from the list. The remaining, distinct separator clusters found by this procedure correspond to the candidate table objects that are passed to the subsequent processing stages.

Table Labels Assignment

To improve precision, the proposed algorithm searches for the presence of predefined keywords (e.g., *table*, *Tab.*) in the close neighborhood of the table candidates identified in the preceding step and, accordingly, marks the table as *labeled* or *unlabeled*. If the labels are required by the current configuration, all unlabeled tables are removed from the set of already found candidates at this stage.

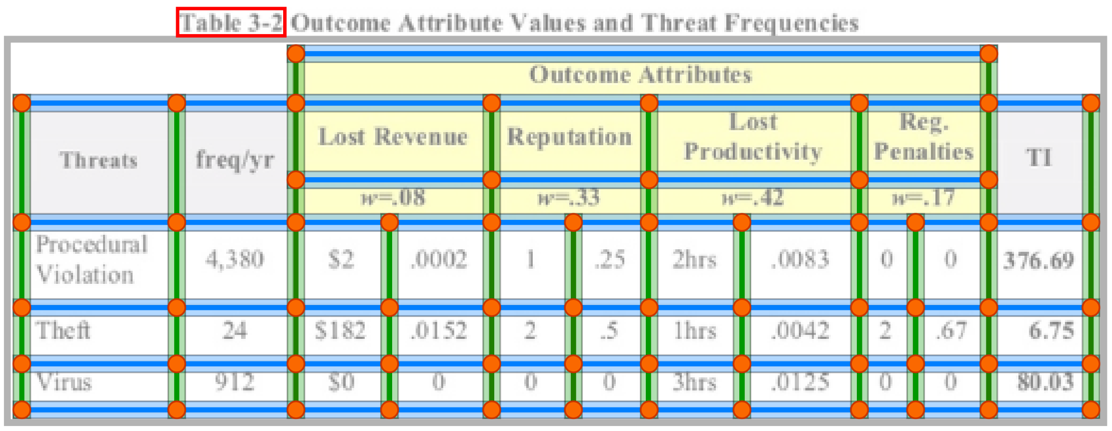


Table 3-2 Outcome Attribute Values and Threat Frequencies

Threats	freq/yr	Outcome Attributes								TI
		Lost Revenue		Reputation		Lost Productivity		Reg. Penalties		
		w=.08		w=.33		w=.42		w=.17		
Procedural Violation	4,380	$2	.0002	1	.25	2hrs	.0083	0	0	376.69
Theft	24	$182	.0152	2	.5	1hrs	.0042	2	.67	6.75
Virus	912	$0	0	0	0	3hrs	.0125	0	0	80.03

Figure 5.4: Separator merging stage of the table recognition method for fully bordered tables. Vertical and horizontal separator regions are marked green and blue, respectively. Orange circles correspond to the intersection points. The red box represents the detected table label.

Table Grid Estimation and Refinement

Subsequently, for each table candidate, a rough grid of cells is derived as follows: Each pair of subsequent vertical and horizontal separators forms a table column or table row region, respectively. The regions of intersection between the column and row boxes define the rough grid of cells.

Note that some cells in the roughly estimated grid need to be refined by merging them with the neighboring cells to recover the cells that span multiple rows or columns. To this end, an approach inspired by the *union-find algorithm* proposed by Hoshen and Kopelman (1976) is employed, as illustrated in Figure 5.5.

Specifically, a raster scan is performed through the rough grid of cells line by line in the left-to-right direction. If the area near the right border of a cell does not overlap any vertical separator assigned to the current separator cluster, the cell is merged with its right neighbor and the algorithm proceeds to the next cell. A margin around the right border of a cell used to calculate the overlap is defined as $m_x = \overline{S}_x$.

Subsequently, the whole procedure is repeated in the top-to-down direction. In this case, the margin $m_y = \overline{S}_y$ is used. Note that the cells that fulfill the merging criterion mentioned above need to have equal row and column spans in the case of the scan in the left-to-right and top-to-down direction, respectively. This additional requirement ensures that all cells retain the rectangular shape after merging is performed.

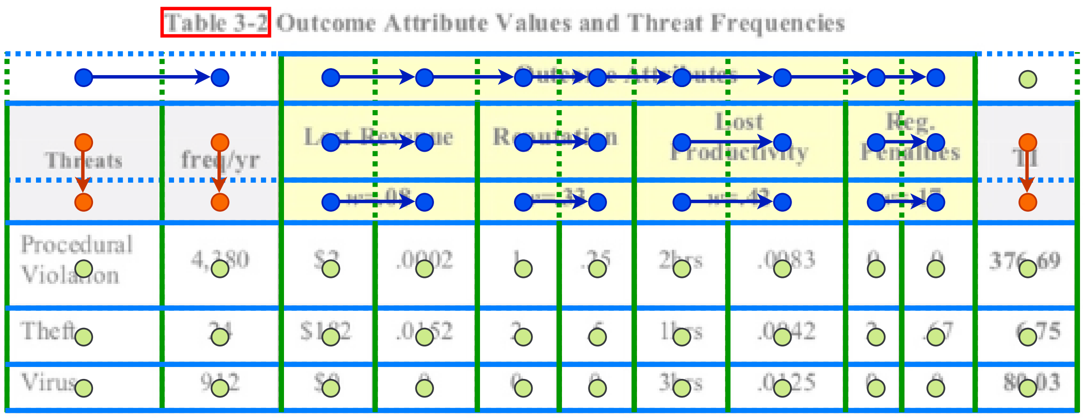


Table 3-2 Outcome Attribute Values and Threat Frequencies

		Outcome Attributes								
Threats	freq/yr	Lost Revenue		Reputation		Lost Productivity		Reg. Penalties		TI
		w=.08		w=.33		w=.42		w=.17		
Procedural Violation	4,380	$2	.0002	1	.25	2hrs	.0083	0	0	376.69
Theft	24	$182	.0152	2	6	1hrs	.0042	2	.67	6.75
Virus	912	$0	0	0	0	3hrs	.0125	0	0	80.03

Figure 5.5: Cell merging stage of the table recognition method for fully bordered tables. Blue and orange circles are the centers of the cells that were merged horizontally and vertically, respectively. Green circles are the centers of fully bordered cells. Arrows show the scanning direction.

Postprocessing

During the postprocessing phase, all textual page regions are assigned to the corresponding table cells based on their overlap ratios; cells that do not contain any assigned page regions are marked as empty. Subsequently, the rows and columns that contain exclusively empty cells are removed from the table.

Finally, all table candidates that have less than a predefined number of rows, columns, and cells are pruned from the list of candidates. Figure 5.6 presents an example of a table recognized by the heuristic for fully bordered tables.

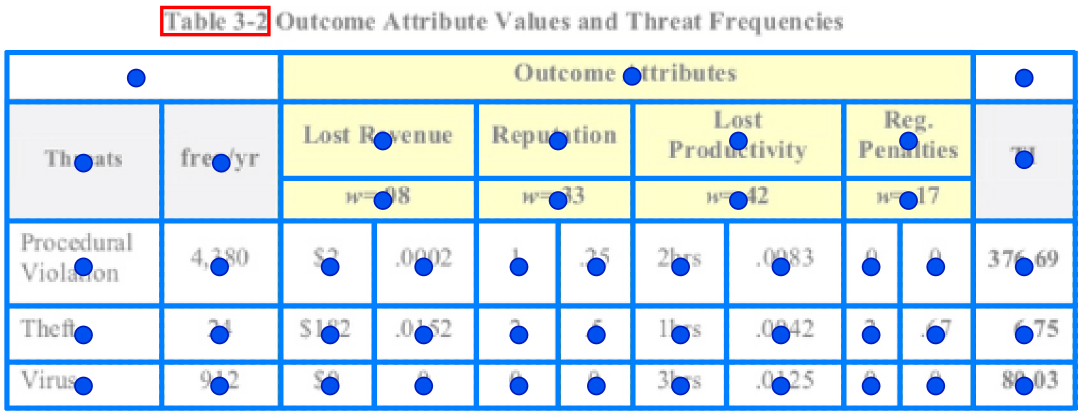


Table 3-2 Outcome Attribute Values and Threat Frequencies

		Outcome Attributes								
Threats	freq/yr	Lost Revenue		Reputation		Lost Productivity		Reg. Penalties		TI
		w=.08		w=.33		w=.42		w=.17		
Procedural Violation	4,380	$2	.0002	1	.25	2hrs	.0083	0	0	376.69
Theft	24	$182	.0152	2	6	1hrs	.0042	2	.67	6.75
Virus	912	$0	0	0	0	3hrs	.0125	0	0	80.03

Figure 5.6: Recognition result obtained by the table recognition method for fully bordered tables. Blue circles represent the centers of the recognized cells.

5.4.4 Recognition of Partially Bordered Tables

Booktabs[12] is a popular LaTeX package used to typeset tables in scientific articles. An example of a table in this format is presented in Figure 5.7. It consists of three main components: a top, middle, and bottom ruling line. The middle ruling line separates the table header and the table body region. In addition, multiple-level header structure can be represented using shorter *cmidrules* that span multiple columns aggregated under the same higher-level header (see Figure 5.11a).

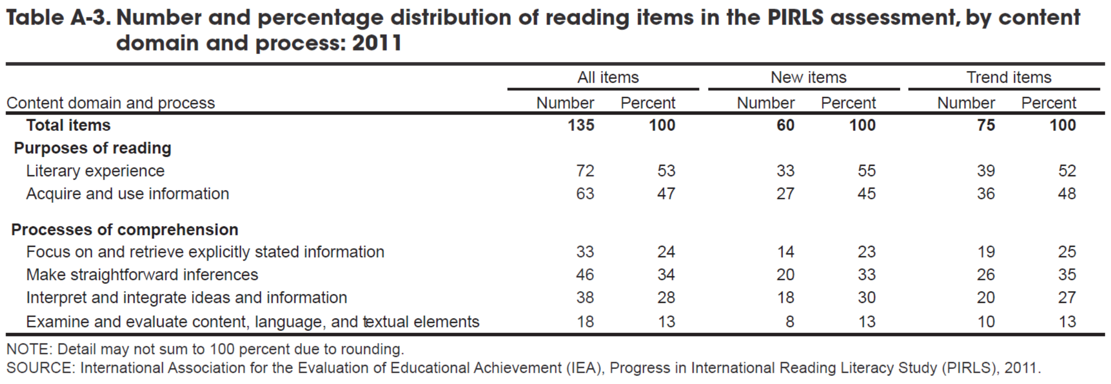


Table A-3. Number and percentage distribution of reading items in the PIRLS assessment, by content domain and process: 2011

	All items		New items		Trend items	
Content domain and process	Number	Percent	Number	Percent	Number	Percent
Total items	**135**	**100**	**60**	**100**	**75**	**100**
Purposes of reading						
Literary experience	72	53	33	55	39	52
Acquire and use information	63	47	27	45	36	48
Processes of comprehension						
Focus on and retrieve explicitly stated information	33	24	14	23	19	25
Make straightforward inferences	46	34	20	33	26	35
Interpret and integrate ideas and information	38	28	18	30	20	27
Examine and evaluate content, language, and textual elements	18	13	8	13	10	13

NOTE: Detail may not sum to 100 percent due to rounding.
SOURCE: International Association for the Evaluation of Educational Achievement (IEA), Progress in International Reading Literacy Study (PIRLS), 2011.

Figure 5.7: An example of a table in *booktabs* format from the `us-021.pdf` file contained in the ICDAR 2013 data set (Göbel et al., 2013).

The presented heuristic for partially bordered tables uses horizontal separators for horizontally oriented pages. As noted in Section 5.4.2, the pages with vertical orientation can be easily handled by the proposed method by swapping horizontal and vertical separators with each other.

Table Region Detection

In the first step, separator filtering is performed. Specifically, all thick lines that are wider than $\overline{S}_y$ are discarded, and the remaining separators are sorted by the top position. Moreover, if multiple separators are located within the margin of $D = 2\overline{S}_y$, only the element with the lowest y-position is kept.

To detect table objects, the triples of consecutive separators are located, for which the difference of their left and right coordinates is lower than $\overline{S}_x$. Moreover, all *cmidrules* that are located between the top and the middle rule are also collected and associated with the corresponding table region, so they can be used to recognize a multiple-level header structure in the subsequent processing step. To this end, they are grouped by their y-position to isolate different levels of the

[12] `https://ctan.org/pkg/booktabs`

header's hierarchy and to separate header rows. Furthermore, label assignment is optionally performed as described in Section 5.4.3.

Table Row Detection

The borders for the rows in the body region of a table are determined using the horizontal profile, which is calculated by projecting all words within the body region of a table, as illustrated in Figure 5.8. The row borders can then be easily estimated by taking center positions of the gaps in the resulting profile.

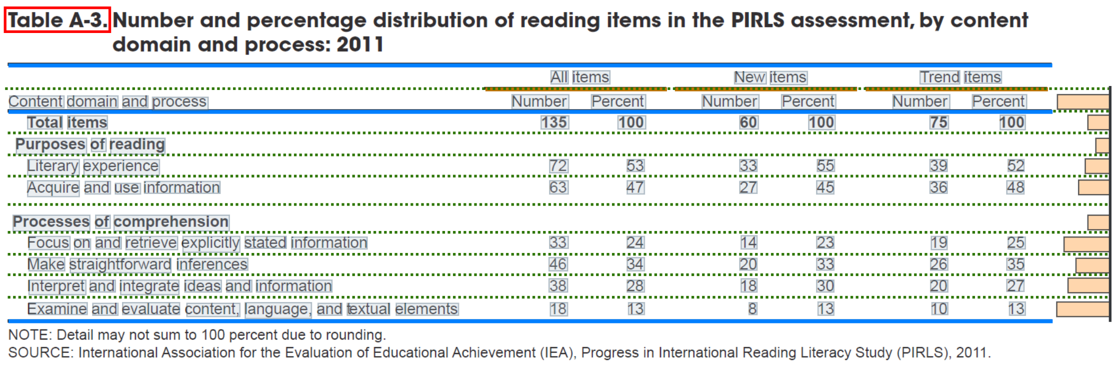


Table A-3. Number and percentage distribution of reading items in the PIRLS assessment, by content domain and process: 2011

Content domain and process	All items Number	All items Percent	New items Number	New items Percent	Trend items Number	Trend items Percent
Total items	**135**	**100**	**60**	**100**	**75**	**100**
Purposes of reading						
Literary experience	72	53	33	55	39	52
Acquire and use information	63	47	27	45	36	48
Processes of comprehension						
Focus on and retrieve explicitly stated information	33	24	14	23	19	25
Make straightforward inferences	46	34	20	33	26	35
Interpret and integrate ideas and information	38	28	18	30	20	27
Examine and evaluate content, language, and textual elements	18	13	8	13	10	13

NOTE: Detail may not sum to 100 percent due to rounding.
SOURCE: International Association for the Evaluation of Educational Achievement (IEA), Progress in International Reading Literacy Study (PIRLS), 2011.

Figure 5.8: Row segmentation process of the proposed table recognition method for partially bordered tables. Blue lines represent the top, middle, and bottom ruling lines. Orange lines depict the *cmidrule* lines. Orange bars to the right correspond to the horizontal profile (running sum of pixels in the text regions in each row). Green dotted lines correspond to the row borders.

Table Column Detection

To recognize the borders between the columns, all page regions within the body region and the lowest-level header row are projected vertically, as illustrated in Figure 5.9. Subsequently, the resulting projection is analyzed to find all gaps between two consecutive table columns with a length above the threshold $\mathcal{D}_{\text{column}}$, which is calculated as follows:

$$\mathcal{D}_{\text{column}} = \mathcal{D}_{\text{page}} \mathcal{H}_{\text{table}} \gamma, \tag{5.2}$$

where $\mathcal{D}_{\text{page}}$ is the median unit distance, i.e., the distance divided by the word height, between two words within a page, $\mathcal{H}_{\text{table}}$ is the mean word height within the table, and γ is a hyperparameter that controls the width of the gaps between the columns.

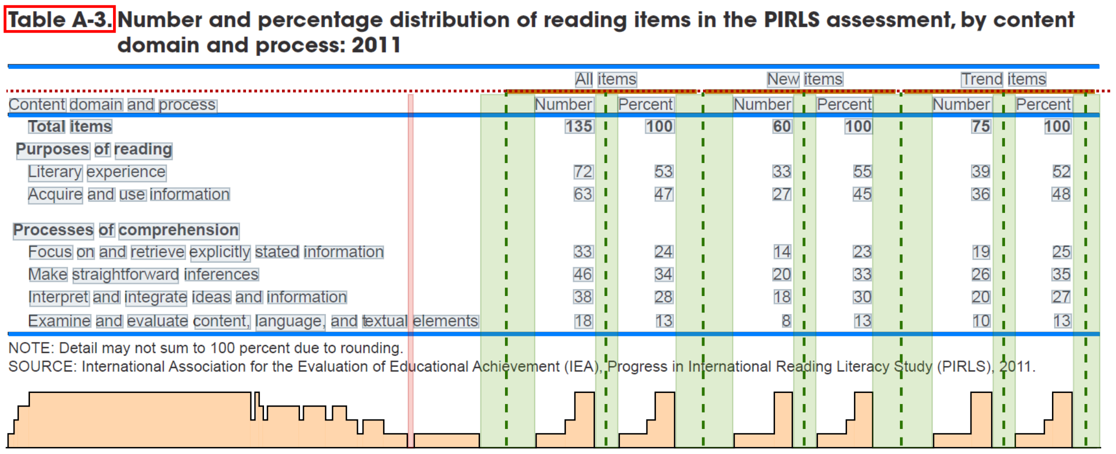


Table A-3. Number and percentage distribution of reading items in the PIRLS assessment, by content domain and process: 2011

	All items		New items		Trend items	
Content domain and process	Number	Percent	Number	Percent	Number	Percent
Total items	**135**	**100**	**60**	**100**	**75**	**100**
Purposes of reading						
Literary experience	72	53	33	55	39	52
Acquire and use information	63	47	27	45	36	48
Processes of comprehension						
Focus on and retrieve explicitly stated information	33	24	14	23	19	25
Make straightforward inferences	46	34	20	33	26	35
Interpret and integrate ideas and information	38	28	18	30	20	27
Examine and evaluate content, language, and textual elements	18	13	8	13	10	13

NOTE: Detail may not sum to 100 percent due to rounding.
SOURCE: International Association for the Evaluation of Educational Achievement (IEA), Progress in International Reading Literacy Study (PIRLS), 2011.

Figure 5.9: Column segmentation process employed by the proposed table recognition method for partially bordered tables. The dotted red line is a border of the lowest-level header. Orange bars at the bottom correspond to the vertical histogram profile, i.e., running sum of pixels in the word regions in each column. The values in the profile are clipped for better visualization. The column gaps that are wider and narrower than $\mathcal{D}_{\text{column}}$ are highlighted in green and red, respectively. Green vertical dotted lines represent the detected column borders.

The center positions of the intervals that satisfy the condition defined above correspond to the column borders. In contrast, all gaps with a length below $\mathcal{D}_{\text{column}}$ correspond to vertically aligned words that form spurious columns. Note that the higher-level headers are excluded, as they contain multiple-column cells that would otherwise distort the calculated vertical projection.

Table Grid Estimation and Refinement

Given the row and column borders calculated in the previous stage, the initial grid of cells is computed from the intersections between the row and the column borders, which results in a partial table segmentation. An example of such a segmentation grid is illustrated in Figure 5.10.

Note that to complement the table segmentation result obtained in the preceding step, the structure of the remaining, higher-level headers needs to be recognized. To this end, the rough grid of cells calculated in the previous step is first extended to the higher-level headers. Subsequently, all cells that intersect the same *cmidrule* segment are merged. Figure 5.11 illustrates the segmentation process of the higher-level header region of a table.

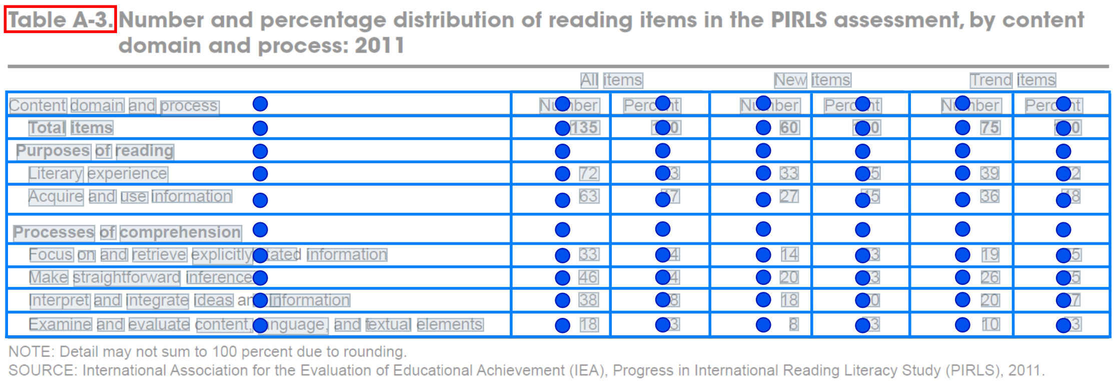


Table A-3. Number and percentage distribution of reading items in the PIRLS assessment, by content domain and process: 2011

	All items		New items		Trend items	
Content domain and process	Number	Percent	Number	Percent	Number	Percent
Total items	135	100	60	100	75	100
Purposes of reading						
Literary experience	72	[illegible]3	33	[illegible]5	39	[illegible]2
Acquire and use information	63	[illegible]7	27	[illegible]5	36	[illegible]8
Processes of comprehension						
Focus on and retrieve explicitly stated information	33	[illegible]4	14	[illegible]3	19	[illegible]5
Make straightforward inferences	46	[illegible]4	20	[illegible]3	26	[illegible]5
Interpret and integrate ideas and information	38	[illegible]8	18	[illegible]0	20	[illegible]7
Examine and evaluate content, language, and textual elements	18	[illegible]3	8	[illegible]3	10	[illegible]3

NOTE: Detail may not sum to 100 percent due to rounding.
SOURCE: International Association for the Evaluation of Educational Achievement (IEA), Progress in International Reading Literacy Study (PIRLS), 2011.

Figure 5.10: The resulting segmentation grid obtained by the proposed table recognition method for partially bordered tables. Blue lines and circles represent the borders and the centers of the cells, respectively. The boxes with gray backgrounds outline the words within the table area.

Postprocessing

Finally, all textual page regions are assigned to the corresponding table cells based on their overlap ratios and the cells that do not contain any assigned page regions are marked as empty.

Subsequently, the rows and columns that contain exclusively empty cells are removed from the table. Moreover, all table candidates that have less than a predefined number of rows, columns, and cells are pruned from the list of candidates.

5.4.5 Proposed Table Interpretation Method

The proposed table interpretation method takes a result of the prior table recognition step as input, i.e., a set of recognized tables T.

Given a set of predefined meanings M and a set of columns C contained in a table $t \in T$, the proposed method first assigns meanings $m \in M$ to the columns $c \in C$. Subsequently, the column-level results are propagated to the individual cells using the following procedure: For a column c that was matched with a meaning m_j, the tuples $x_{i,j}$ containing the target information are constructed by associating the data cells in the body part of the column c with the meaning m_j, where i and j are the index of a table row and a meaning that was matched with the column c, respectively.

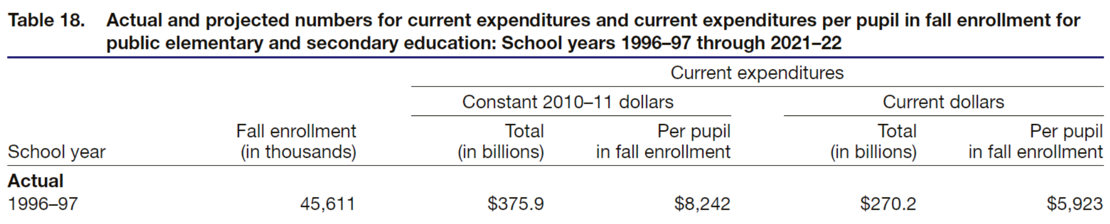


Table 18. Actual and projected numbers for current expenditures and current expenditures per pupil in fall enrollment for public elementary and secondary education: School years 1996–97 through 2021–22

		Current expenditures			
		Constant 2010–11 dollars		Current dollars	
School year	Fall enrollment (in thousands)	Total (in billions)	Per pupil in fall enrollment	Total (in billions)	Per pupil in fall enrollment
Actual					
1996–97	45,611	$375.9	$8,242	$270.2	$5,923

(a) Input Image

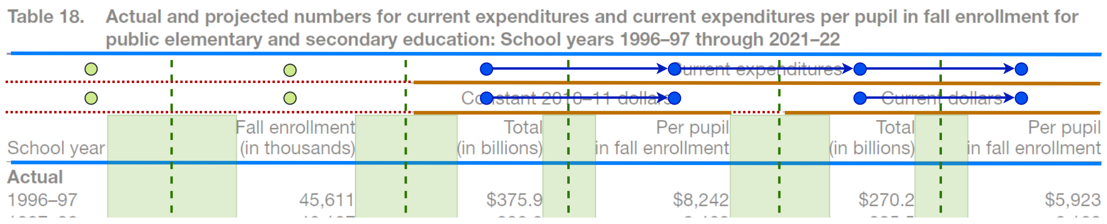


Table 18. Actual and projected numbers for current expenditures and current expenditures per pupil in fall enrollment for public elementary and secondary education: School years 1996–97 through 2021–22

		Current expenditures			
		Constant 2010–11 dollars		Current dollars	
School year	Fall enrollment (in thousands)	Total (in billions)	Per pupil in fall enrollment	Total (in billions)	Per pupil in fall enrollment
Actual					
1996–97	45,611	$375.9	$8,242	$270.2	$5,923

(b) Header Cell Merging

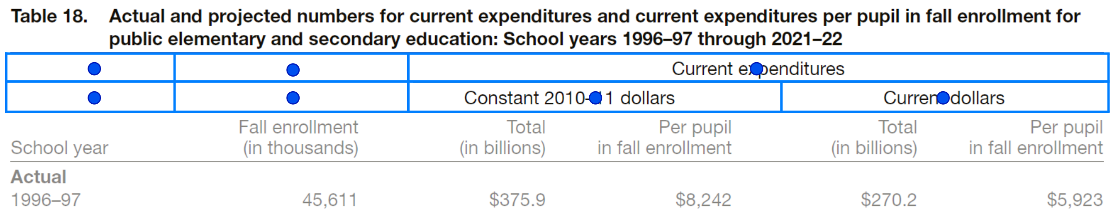


Table 18. Actual and projected numbers for current expenditures and current expenditures per pupil in fall enrollment for public elementary and secondary education: School years 1996–97 through 2021–22

		Current expenditures			
		Constant 2010–11 dollars		Current dollars	
School year	Fall enrollment (in thousands)	Total (in billions)	Per pupil in fall enrollment	Total (in billions)	Per pupil in fall enrollment
Actual					
1996–97	45,611	$375.9	$8,242	$270.2	$5,923

(c) Final Header Segmentation

Figure 5.11: Higher-level header segmentation method employed in the proposed table recognition method for partially bordered tables. **(a)** The top part of a table extracted from the `us-018.pdf` file from the ICDAR 2013 benchmark (Göbel et al., 2013). **(b)** Header cell merging. Orange lines correspond to the *cmidrule* lines. Green areas and lines represent column whitespaces and borders, respectively. Blue circles are the centers of the cells intersecting a *cmidrule* line. The cells that intersect the same *cmidrule* line are merged. In contrast, other cells (marked with green circles) remain unchanged. **(c)** Header segmentation. Blue lines and circles correspond to the borders and the centers of the cells in the final grid, respectively.

Affinity Score Computation

The rules for assigning the columns to the meanings are established as follows. For each meaning m, a customized set of affinity rules that describe a column that is likely to be matched with m is defined:

(1) *Title Keyword Score*: approximate string matching between the title of a column and the predefined keywords.

(2) *Title RegEx Score*: exact matching of the title of a column with customized regular expressions.

(3) *Data Type Score*: exact matching of the content of the cells in a column with regular expressions for predefined types (e.g., integer, date, etc.).

(4) *Content RegEx Score*: exact matching of the content of the cells in a column with customized regular expressions.

Approximate string matching corresponds to the Levenshtein distance (Levenshtein, 1966) calculated between two strings and divided by the length of the longer string. The exact RegEx score returns 1.0 if the matching succeeds and 0.0 otherwise. Moreover, note that the content and data type scores are averaged over the scores for the cells in the corresponding column.

The final affinity score S for a column c with a meaning m is computed as presented in Equation (5.3):

$$S(c,m) = \frac{w_{\mathrm{c}} \max(S_{\mathrm{c}}^{\mathrm{Rx}}, S_{\mathrm{c}}^{\mathrm{DT}}) + w_{\mathrm{t}} \max(S_{\mathrm{t}}^{\mathrm{Rx}}, S_{\mathrm{t}}^{\mathrm{KW}})}{w_{\mathrm{c}} + w_{\mathrm{t}}}, \tag{5.3}$$

where w_{t} and w_{c} are the weights of the title and the content property groups, respectively. $S_{\mathrm{c}}^{\mathrm{Rx}}$ and $S_{\mathrm{c}}^{\mathrm{DT}}$ are the affinity scores of the content RegEx and the data type, respectively. $S_{\mathrm{t}}^{\mathrm{Rx}}$ and $S_{\mathrm{t}}^{\mathrm{KW}}$ correspond to the scores of the title RegEx and the approximate string matching with the keywords, respectively.

Note that the sum of weights must be a positive number. Moreover, if a particular rule is not defined for a meaning, the corresponding score is set to zero. All rules are defined in a configuration file, as presented in an example in Figure 5.12.

Matching Table Columns with Semantic Concepts

To perform the matching between the meanings and the columns in a table, a weighted bipartite graph is created, as illustrated in Figure 5.13b. In this graph, two sets of vertices are defined — representing the meanings on one side and the

```
[
  { "id": "compound",
    "keywords": ["Compound", "compd", "Comp.", "cpd"],
    "datatype": "string",
    "weightTitle": 1.0,
    "weightContent": 0.0,
    "minAffinityScore": 0.5
  },
  { "id": "hdac1_gene",
    "keywords": ["HDAC1"],
    "titleRegex": "^HDAC[-]{0,1}1[^\\d].*$",
    "datatype": ["double", "range", "integer"],
    "weightTitle": 1.0,
    "weightContent": 0.0,
    "minAffinityScore": 0.85
  },
  { "id": "hdac6_gene",
    "keywords": ["HDAC6"],
    "titleRegex": "^HDAC[-]{0,1}6[^\\d].*$",
    "datatype": ["double", "range", "integer"],
    "weightTitle": 1.0,
    "weightContent": 0.0,
    "minAffinityScore": 0.85
  }
]
```

Figure 5.12: Configuration file used by the proposed table interpretation method. In the presented example, the meanings COMPOUND, HDAC1, and HDAC6 GENE are defined, as well as the rules for matching table columns to these meanings. The file is stored in JavaScript Object Notation (JSON) format.

columns on the other side. Moreover, all columns are connected with all meanings with an edge that is weighted by the affinity score $S(c, m)$ that specifies how likely a column c matches with a certain meaning m. Note that the connections that do not reach a predefined required minimum affinity value S_{min} are pruned.

To find the best assignment of the columns to the meanings, maximum weight matching (Edmonds, 1965) on the created bipartite graph is performed. Finally, the tuples $x_{i,j}$ are extracted, where i is an index of a row in the body part of the table, and j is the index of a matched meaning (see Figure 5.13c).

5.5 Proposed Hybrid Table Extraction Method

In this section, the *hybrid* variant of the proposed table extraction system is presented (Figure 5.14). It extends the *basic* variant described in Section 5.4.

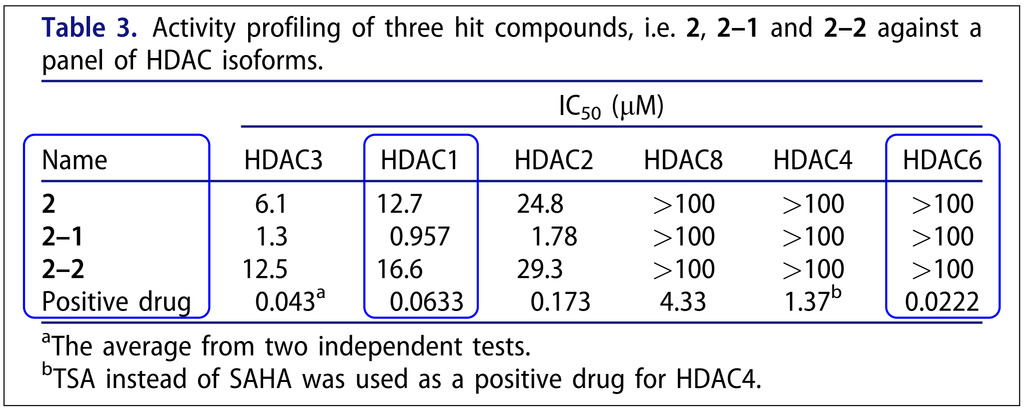
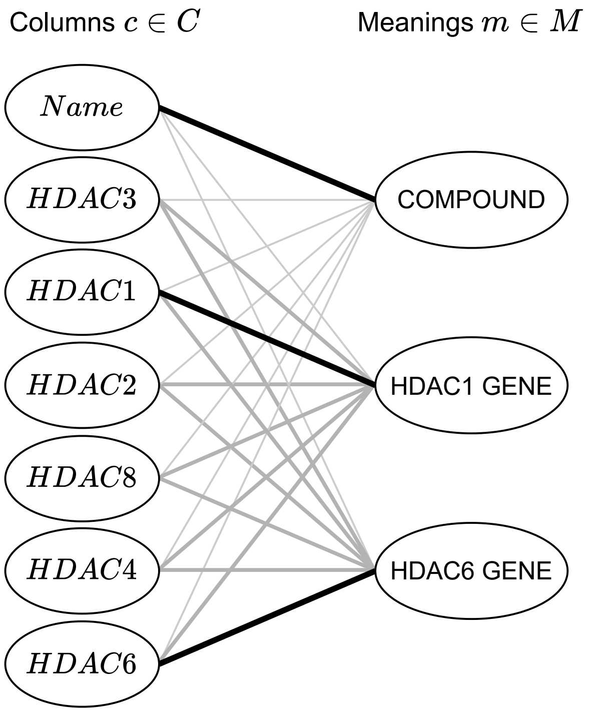


Table 3. Activity profiling of three hit compounds, i.e. **2**, **2–1** and **2–2** against a panel of HDAC isoforms.

	IC_{50} (μM)					
Name	HDAC3	HDAC1	HDAC2	HDAC8	HDAC4	HDAC6
2	6.1	12.7	24.8	>100	>100	>100
2–1	1.3	0.957	1.78	>100	>100	>100
2–2	12.5	16.6	29.3	>100	>100	>100
Positive drug	0.043[a]	0.0633	0.173	4.33	1.37[b]	0.0222

[a]The average from two independent tests.
[b]TSA instead of SAHA was used as a positive drug for HDAC4.

(a) Input Table



(b) Interpretation Graph

```
[
  {
    "compound": "2",
    "hdac1_ic50": "12.7",
    "hdac6_ic50": ">100"
  },
  {
    "compound": "2-1",
    "hdac1_ic50": "0.957",
    "hdac6_ic50": ">100"
  },
  {
    "compound": "2-2",
    "hdac1_ic50": "16.6",
    "hdac6_ic50": ">100"
  },
  {
    "compound": "Positive drug",
    "hdac1_ic50": "0.0633",
    "hdac6_ic50": "0.0222"
  }
]
```

(c) Extracted Tuples

Figure 5.13: Illustration of the proposed table interpretation method: **(a)** A table extracted from Xia et al. (2018), which contains the inhibitory activity of some representative compounds toward the histone deacetylase (HDAC) gene. The columns corresponding to the defined meanings are marked with blue boxes. **(b)** Table interpretation graph: Columns $c \in C$ are mapped to the meanings $m \in M$. For each mapping, an affinity value is calculated, symbolized by the thickness of the lines. **(c)** The extracted tuples that represent the inhibitory activity of each compound toward the HDAC1 and the HDAC6 gene. The resulting file is stored in JSON format.

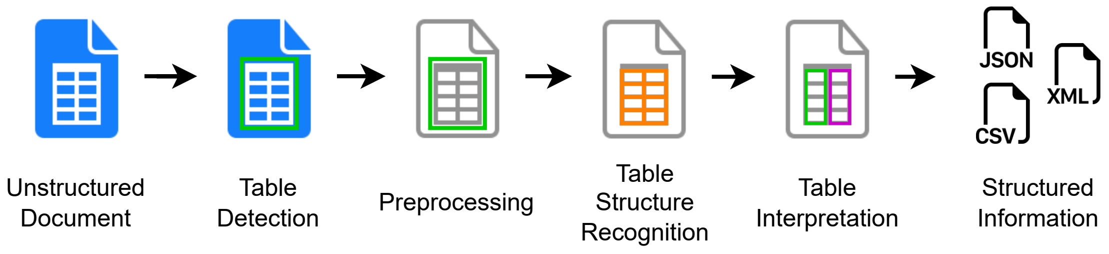


Figure 5.14: The hybrid variant of the proposed IE system. An unstructured document, either an image or a PDF file, is given as input. Table detection locates all tables within an input document. Preprocessing is performed prior to TSR, which recognizes the building blocks of a table — rows, columns, and individual table cells. Table interpretation links the extracted structural elements with predefined semantic concepts. As a result, the layout and the semantic interpretation of a table is written in structured format.

As before, we assume that an unstructured document, either an image or a PDF file, is given as input to the system. Firstly, *table detection* is performed to locate all tables within the input document (Section 5.5.1). Secondly, each detected table is cropped from the input document and passed through the *preprocessing* module to prepare it for subsequent analysis, as described in Section 5.4.1. Thirdly, for each detected table individually, TSR is executed to recognize the building blocks of a table — rows, columns, and cells (Section 5.5.2). Finally, *table interpretation* is carried out to link the extracted structural elements with predefined semantic concepts (Section 5.4.5).

The presented system combines a deep learning-based table detection module, rule-based TSR heuristics, and a graph-based table interpretation method. To perform TSR, the heuristics described in Sections 5.4.3 and 5.4.4 are adapted to handle documents containing individual table objects (Section 5.5.2). For table interpretation, the method described in Section 5.4.5 is employed.

5.5.1 Table Detection

Table detection aims to locate all tables within an input document (Section 5.2.1). Recent advances in deep learning-based object recognition (J. Wang et al., 2021; S. Xie et al., 2017) allow to perform a highly accurate and reliable table detection process. Therefore, the hybrid variant of the proposed system exploits an existing deep learning-based table localization method combined with a heuristic-based TSR module, resulting in an efficient, hybrid table recognition approach.

The detection method is required to take either an image or a PDF file as input and to return a list of bounding boxes, each corresponding to a single table object.

If the confidence values for each detection result are also provided, they can be exploited by the proposed method to perform additional filtering, as described in Section 5.6.2. In general, the choice of the detection method is arbitrary, as long as the aforementioned requirements are met.

Using the results provided by the table detection component, all identified tables are cropped from the original input document and passed to the preprocessing module as either images or PDF files, depending on the format of the original document.

5.5.2 Table Structure Recognition

The table recognition module described in Section 5.4 takes a semi-structured representation of a whole document as input and performs table detection and structure recognition in one step. In contrast, the approach described in this section performs table detection and TSR separately. Therefore, the input to the TSR component constitutes a document containing a single table object together with its semi-structured representation.

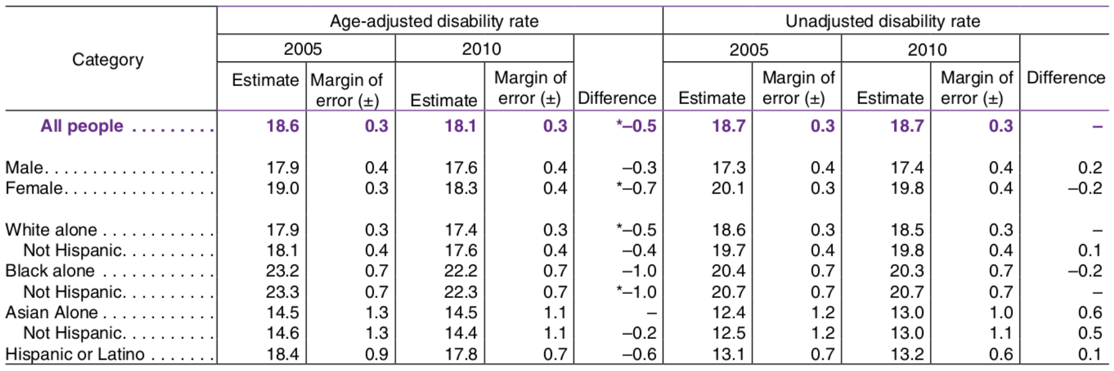
To perform TSR, the heuristics described in Sections 5.4.3 and 5.4.4 need to be adapted to this scenario. The first, straightforward modification is that table labels are not used for filtering table candidates (Section 5.4.3) because the table detection module already delivers fairly accurate detection results. Further adaptations and improvements, designed specifically for each heuristic, are described in the remaining part of this section.

Recognition of Fully Bordered Tables

This TSR method employs the heuristic presented in Section 5.4.3 with the modifications described in the following paragraphs.

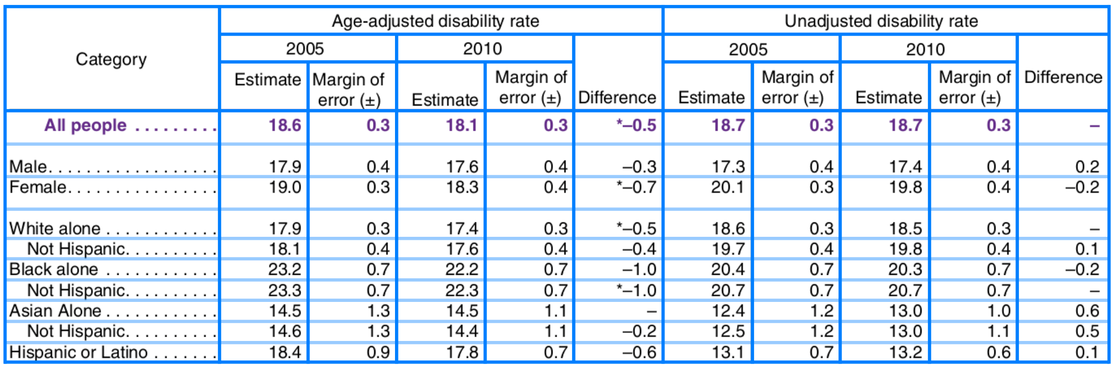
Dynamic Margin for Separator Merging The margins employed in the separator merging procedure (Section 5.4.3) are calculated as follows: $\delta_x = \max(5, \overline{S}_x/2)$ and $\delta_y = \max(5, \overline{S}_y/2)$. This makes the method more resistant to different resolutions of the input document, preserving the minimum margin of five pixels used in the basic approach.

Filtering Hybrid-Layout Tables In the postprocessing stage (Section 5.4.3), additional filtering is performed to discard the tables that predominantly exhibit a bordered layout but also contain many rows that are separated by whitespaces instead of solid separators, as such tables would preferably be recognized using the heuristic for partially bordered tables (see Figure 5.15). For each table, the ratio H_{ratio} of the highest row to the median row height is calculated and all tables with H_{ratio} greater than a predefined threshold $H_{\text{ratio}}^{\text{max}}$ are discarded.



Category	Age-adjusted disability rate					Unadjusted disability rate				
	2005		2010			2005		2010		
	Estimate	Margin of error (±)	Estimate	Margin of error (±)	Difference	Estimate	Margin of error (±)	Estimate	Margin of error (±)	Difference
All people	**18.6**	**0.3**	**18.1**	**0.3**	***–0.5**	**18.7**	**0.3**	**18.7**	**0.3**	**–**
Male..................	17.9	0.4	17.6	0.4	–0.3	17.3	0.4	17.4	0.4	0.2
Female................	19.0	0.3	18.3	0.4	*–0.7	20.1	0.3	19.8	0.4	–0.2
White alone	17.9	0.3	17.4	0.3	*–0.5	18.6	0.3	18.5	0.3	–
Not Hispanic..........	18.1	0.4	17.6	0.4	–0.4	19.7	0.4	19.8	0.4	0.1
Black alone	23.2	0.7	22.2	0.7	–1.0	20.4	0.7	20.3	0.7	–0.2
Not Hispanic..........	23.3	0.7	22.3	0.7	*–1.0	20.7	0.7	20.7	0.7	–
Asian Alone	14.5	1.3	14.5	1.1	–	12.4	1.2	13.0	1.0	0.6
Not Hispanic..........	14.6	1.3	14.4	1.1	–0.2	12.5	1.2	13.0	1.1	0.5
Hispanic or Latino	18.4	0.9	17.8	0.7	–0.6	13.1	0.7	13.2	0.6	0.1

(a) Input Image



Category	Age-adjusted disability rate					Unadjusted disability rate				
	2005		2010			2005		2010		
	Estimate	Margin of error (±)	Estimate	Margin of error (±)	Difference	Estimate	Margin of error (±)	Estimate	Margin of error (±)	Difference
All people	**18.6**	**0.3**	**18.1**	**0.3**	***–0.5**	**18.7**	**0.3**	**18.7**	**0.3**	**–**
Male..................	17.9	0.4	17.6	0.4	–0.3	17.3	0.4	17.4	0.4	0.2
Female................	19.0	0.3	18.3	0.4	*–0.7	20.1	0.3	19.8	0.4	–0.2
White alone	17.9	0.3	17.4	0.3	*–0.5	18.6	0.3	18.5	0.3	–
Not Hispanic..........	18.1	0.4	17.6	0.4	–0.4	19.7	0.4	19.8	0.4	0.1
Black alone	23.2	0.7	22.2	0.7	–1.0	20.4	0.7	20.3	0.7	–0.2
Not Hispanic..........	23.3	0.7	22.3	0.7	*–1.0	20.7	0.7	20.7	0.7	–
Asian Alone	14.5	1.3	14.5	1.1	–	12.4	1.2	13.0	1.0	0.6
Not Hispanic..........	14.6	1.3	14.4	1.1	–0.2	12.5	1.2	13.0	1.1	0.5
Hispanic or Latino	18.4	0.9	17.8	0.7	–0.6	13.1	0.7	13.2	0.6	0.1

(b) Recognition Result

Figure 5.15: An example of a table cropped from the `us-001.jpg` file contained in the ICDAR 2013 benchmark (Göbel et al., 2013). Solid blue lines represent the borders between the cells that were detected by the TSR heuristic for fully bordered tables. In contrast, light blue lines correspond to the row borders that are not outlined with solid ruling lines and therefore could not be recognized by this method.

Recognition of Partially Bordered and Borderless Tables

This TSR method employs the heuristic presented in Section 5.4.4 with the modifications described in the following paragraphs.

Adding Virtual Ruling Lines The recognition of the tables that do not strictly follow the booktabs format, e.g., by missing a top or a bottom ruling line, is facilitated by adding a virtual top and bottom ruling lines at the top and the bottom of the input document, respectively. These ruling lines are stretched across the entire width of the image and used in the subsequent table region detection procedure (Section 5.4.4). Moreover, if the detection process does not output any valid table candidate, an additional virtual ruling line is added at the position $-\overline{S}_y$, and the searching process is repeated. This step ensures that at least one table candidate is found, even in the case of *borderless* tables.

Filtering Narrow Tables An additional filtering step is added, where all candidates narrower than 90% of the image width are discarded. Figure 5.16 illustrates a case that benefits from adding virtual ruling lines and filtering narrow tables.

Merging Overlapping Table Candidates In the case of the tables that use solid separators for the separation of rows, the basic approach outputs multiple overlapping candidates. Therefore, if the vertical overlap between two table candidates is greater than $\overline{S}_y$, these candidates are merged, i.e., the top and the middle ruling line with a lower y-position as well as the bottom line with a higher y-position are retained in the merged table candidate, as illustrated in Figure 5.17. This procedure is performed prior to the table row detection step (Section 5.4.4).

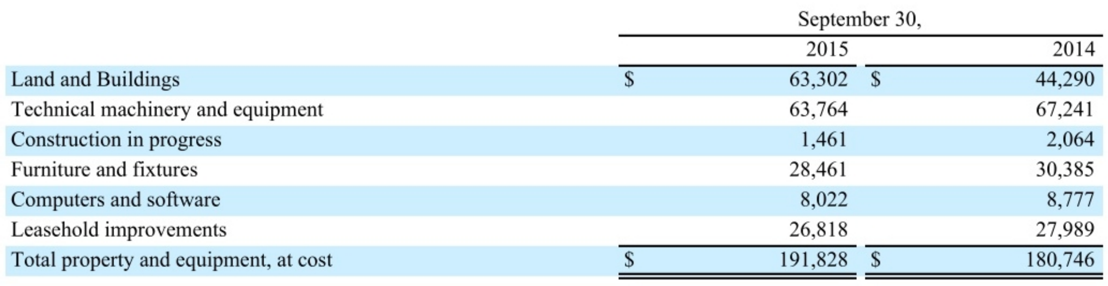
Estimation of the Threshold for Column Separation The threshold used to separate table columns is estimated as follows:

$$\mathcal{D}_{\text{column}} = \gamma \overline{S}_x, \tag{5.4}$$

where γ is a hyperparameter (Section 5.4.4). The average character size $\overline{S}_x$ within the table is used instead of the median unit distance between the words on a page (Equation (5.2)) because the full-page content cannot be exploited in this scenario.

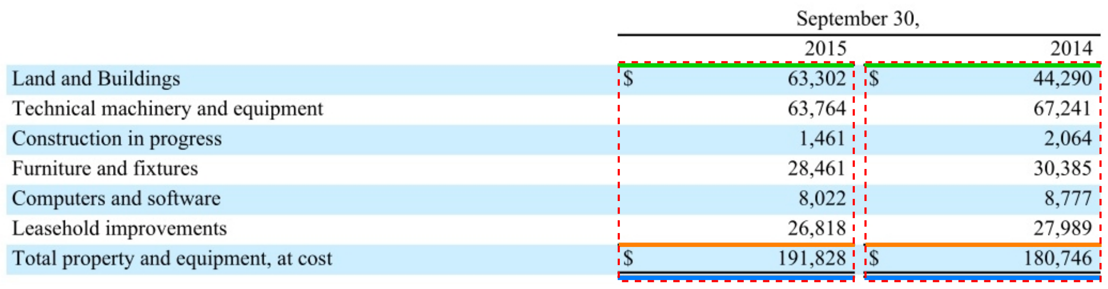
5.6 Experimental Setup - Table Recognition

In this section, the experimental setup of the evaluation presented in Section 5.7 is described, where the proposed hybrid table extraction system (Section 5.5) is examined using two widely adopted table recognition benchmarks. In both cases, the CTR process is evaluated, i.e., end-to-end table detection and structure recognition.



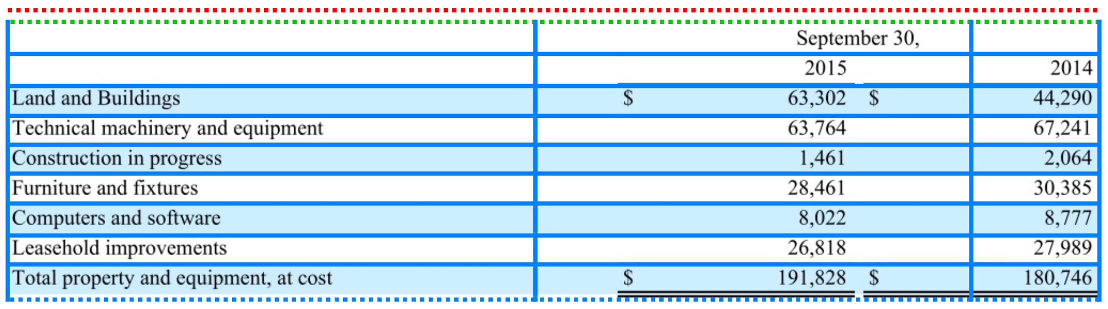
	September 30,			
		2015		2014
Land and Buildings	$	63,302	$	44,290
Technical machinery and equipment		63,764		67,241
Construction in progress		1,461		2,064
Furniture and fixtures		28,461		30,385
Computers and software		8,022		8,777
Leasehold improvements		26,818		27,989
Total property and equipment, at cost	$	191,828	$	180,746

(a) Input Image



	September 30,			
		2015		2014
Land and Buildings	$	63,302	$	44,290
Technical machinery and equipment		63,764		67,241
Construction in progress		1,461		2,064
Furniture and fixtures		28,461		30,385
Computers and software		8,022		8,777
Leasehold improvements		26,818		27,989
Total property and equipment, at cost	$	191,828	$	180,746

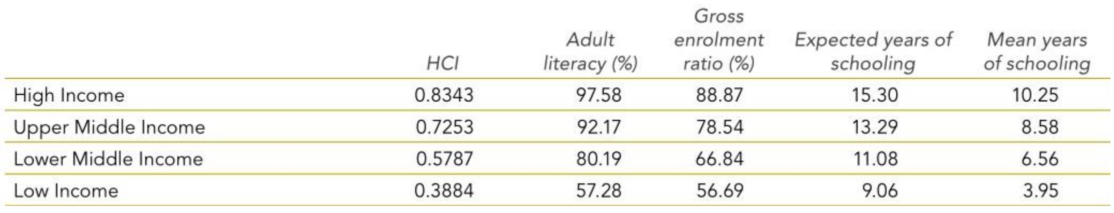
(b) Initial Detection Result Without Filtering



	September 30,			
		2015		2014
Land and Buildings	$	63,302	$	44,290
Technical machinery and equipment		63,764		67,241
Construction in progress		1,461		2,064
Furniture and fixtures		28,461		30,385
Computers and software		8,022		8,777
Leasehold improvements		26,818		27,989
Total property and equipment, at cost	$	191,828	$	180,746

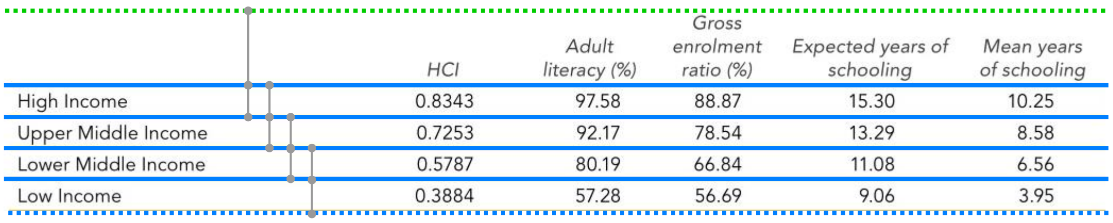
(c) Final Results After Filtering

Figure 5.16: Filtering based on the table width employed by the TSR heuristic for partially bordered and borderless tables. **(a)** An example of a table cropped from the `cTDaR_t10005.jpg` file contained in the ICDAR 2019 benchmark (L. Gao et al., 2019). **(b)** An initial result before the filtering — two spurious candidates were identified. Green, orange, and blue lines correspond to the top, middle, and bottom ruling lines, respectively. **(c)** The result after filtering. Dotted green and blue lines correspond to the virtual top and bottom ruling lines, respectively. Dotted red line is the virtual ruling line added above the top ruling line. Note that the row between two virtual ruling lines at the top is discarded as it does not contain any textual content.



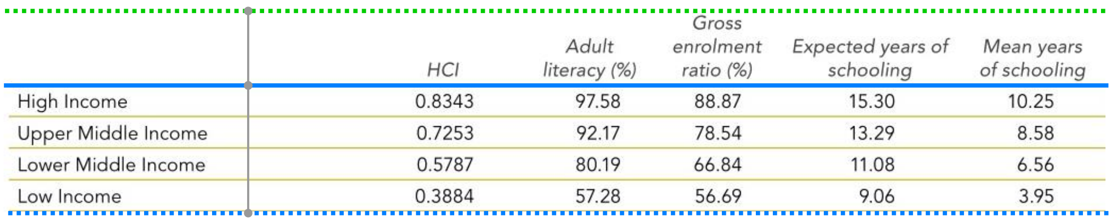
	HCI	Adult literacy (%)	Gross enrolment ratio (%)	Expected years of schooling	Mean years of schooling
High Income	0.8343	97.58	88.87	15.30	10.25
Upper Middle Income	0.7253	92.17	78.54	13.29	8.58
Lower Middle Income	0.5787	80.19	66.84	11.08	6.56
Low Income	0.3884	57.28	56.69	9.06	3.95

(a) Input Image



	HCI	Adult literacy (%)	Gross enrolment ratio (%)	Expected years of schooling	Mean years of schooling
High Income	0.8343	97.58	88.87	15.30	10.25
Upper Middle Income	0.7253	92.17	78.54	13.29	8.58
Lower Middle Income	0.5787	80.19	66.84	11.08	6.56
Low Income	0.3884	57.28	56.69	9.06	3.95

(b) Table Candidates Before Merging



	HCI	Adult literacy (%)	Gross enrolment ratio (%)	Expected years of schooling	Mean years of schooling
High Income	0.8343	97.58	88.87	15.30	10.25
Upper Middle Income	0.7253	92.17	78.54	13.29	8.58
Lower Middle Income	0.5787	80.19	66.84	11.08	6.56
Low Income	0.3884	57.28	56.69	9.06	3.95

(c) Table Candidates After Merging

Figure 5.17: Merging overlapping table candidates, as employed by the TSR heuristic for partially bordered and borderless tables. **(a)** An example of a table cropped from the `cTDaR_t10058.jpg` file contained in the ICDAR 2019 benchmark (L. Gao et al., 2019). **(b)** and **(c)** Overlapping tables. Each triple of consecutive separators, marked with gray lines, represents one table candidate. Using solid lines as row separators causes that a common line is included in the subsequent candidates. Merging the overlapping elements allows to mitigate this problem. Dotted green and blue lines correspond to virtual top and bottom ruling lines, respectively. Solid blue lines represent the remaining solid separators.

5.6.1 Data Sets

The benchmarks employed in the experiments were released by the organizers of the table recognition competitions that were held at the International Conference on Document Analysis and Recognition.

The *ICDAR 2013 Table Competition* data set (Göbel et al., 2013) contains born-digital business and government PDF documents with 156 tables. Ground-truth annotations for both table detection and TSR tasks are available.

The *ICDAR 2019 Table Detection and Recognition* data set (cTDaR; L. Gao et al., 2019) is a collection of modern and archival document images. Only the former part was used in the experiments in this section, as the latter consists of handwritten documents, and the analysis of hand-drawn tables is outside the scope of this work. The *track B2* in this competition was chosen for evaluation as it corresponds to the CTR process, which is the most challenging task. The test set of the aforementioned track consists of 100 images of scanned document pages, each containing at least one table.

5.6.2 Table Detection Setup

In the case of the ICDAR 2013 data set, all pages of a PDF document are first rendered as images with the resolution of 300 dots per inch (DPI) and the detection is performed for each rendered image separately. In the case of the ICDAR 2019 benchmark, the original images are used as input to the detection model.

The hybrid variant of the proposed system employs two previously released table detection models that are described in the following paragraphs. Nevertheless, other models can readily be used instead (Section 5.5.1).

Domain-Specific Table Detection Model

The first variant is the table detection model released by Prasad et al. (2020). Their *CascadeTabNet* method uses an instance segmentation technique and performs pixel-level table identification. In the experiment in Section 5.7.1, a model fine-tuned on the ICDAR 2013 benchmark is used and in Section 5.7.2, the model tuned on the ICDAR 2019 data set is employed. Please refer to Prasad et al. (2020) for the details about the architecture of the table detection model, the composition of the data used for training, and the employed training setup. Hereinafter we refer to this model as the *domain-specific* table detection model.

General-Purpose Table Detection Model

In contrast, the second variant of the hybrid table recognition system employs the table detection model proposed by M. Li et al. (2020), which is based on the *Faster R-CNN* architecture (Ren et al., 2015) with the *ResNeXt-152* model as backbone (S. Xie et al., 2017). Their model was pretrained on the *ImageNet* data set (J. Deng et al., 2009) and fine-tuned on the *TableBank* data (M. Li et al., 2020), which contains a large number of Word and LaTeX documents crawled from the internet. Note that this model was not fine-tuned on the examples from the benchmarks employed for evaluation. Therefore, we refer to this model as a *general-purpose* table detection model.

Filtering Rules

Note that both employed table detection models take an image as input and return, for each detected table, a bounding box and a confidence value. All detection results with a confidence lower than 0.85 and 0.1 are discarded, respectively, in the case of the domain-specific and the general-purpose table detector. Moreover, if some detection results overlap with each other by more than 50%, only the result with a higher confidence value is kept.

5.6.3 TSR Setup

All detected tables are cropped from the input documents based on the returned bounding box coordinates and fed to the preprocessing module, followed by the TSR component, one table at a time. In the case that the input document is in PDF format, the PyPDF2 library[13] is used to crop a region from a PDF file. Therefore, the preprocessing module can extract the text embedded in the PDF files, which is essential to obtain competitive results on the ICDAR 2013 benchmark.

Hyperparameters

The values of the hyperparameters are presented in Table A.5. They were empirically estimated based on the results on a practice data set from ICDAR 2013 Table Competition that consists of 58 PDF documents and the data from the remaining tracks in ICDAR 2019 Table Competition.

[13] `https://pypi.org/project/PyPDF2`

5.6.4 Postprocessing

In the case of the ICDAR 2019 benchmark, the results for all tables on a page are gathered to produce the output XML file in a format[14] that is supported by this competition. Similarly, in the ICDAR 2013 setup, the results of TSR from all pages are gathered to produce the final XML file in a format[15] exploited by the evaluation tools employed in this competition.

5.7 Experimental Results - Table Recognition

In this section, the hybrid variant of the proposed table extraction system (Section 5.5) is thoroughly examined using two widely adopted table recognition benchmarks (Section 5.6.1).

5.7.1 ICDAR 2013 Experiment

In this section, the results of the evaluation performed on the benchmark from ICDAR 2013 Table Competition[16] are presented.

Evaluation Setup

The official evaluation tool written in Java programming language was released by the organizers of this competition.[17] It takes a PDF file and two XML files, with the ground-truth annotations and the recognition results, respectively, as input and produces the resulting precision, recall, and F_1 scores for each ground-truth table in the input document. Moreover, it also outputs the number of relations in all false-positively recognized tables. Unfortunately, the document-level scores used as the main metric in this competition have to be computed manually by accumulating the results from each file in this data set. Specifically, according to the evaluation setup described in (Göbel et al., 2013), the precision and recall scores are first computed for each document separately, and then the average scores are calculated based on the document-level scores.

To facilitate the evaluation process, in this work, a Python wrapper for the competition's evaluation tool was developed and released publicly to promote

[14] https://cndplab-founder.github.io/cTDaR2019/dataset-description.html
[15] https://roundtrippdf.com/en/data-extraction/dataset-format
[16] https://roundtrippdf.com/en/data-extraction/icdar-2013-table-competition
[17] https://roundtrippdf.com/en/data-extraction/table-recognition-dataset-tools

reproducibility and to enable fair comparison of research results obtained on this benchmark.[18] The wrapper script parses the output produced by the official evaluation tool and accumulates the document-level scores to return the final, per document averages — precision, recall, and F_1 score. Moreover, the wrapper script also includes the adjacency relations from the false-positively detected tables to give a better perspective on the actual performance of the table recognition approaches. Furthermore, the script utilizes alternative ground-truth annotations prepared by the organizers for several documents in this data sets.

Evaluation Results

Table 5.1 compares the results of the proposed method with the state-of-the-art approaches on the ICDAR 2013 data set. Note that only the prior work that reported the results of the CTR process is included. Moreover, the methods that used a subset of the data for evaluation are also excluded.

The F_1 score achieved by the proposed method is better than all previously reported results except for the commercial FineReader approach that won the original competition. Comparing the results of two variants of the proposed hybrid approach that employed either the general-purpose or the domain-specific table detection model (Section 5.6.2) we can see a clear advantage of the latter. Nevertheless, the general-purpose variant is still very competitive, outperforming other methods, except for the FineReader engine.

Table 5.1: Evaluation results on the ICDAR 2013 benchmark. The precision, recall, and F_1 score (per document averages) for the CTR process are reported.

Method	Precision	Recall	F_1
FineReader v11 (Göbel et al., 2013)	0.8710	**0.8835**	**0.8772**
Proposed[a]	**0.8714**	0.8468	0.8589
Proposed[b]	0.8483	0.8397	0.8439
OmniPage 18 (Göbel et al., 2013)	0.8460	0.8380	0.8420
Nurminen (Göbel et al., 2013)	0.8693	0.8078	0.8374
TabbyPDF (Shigarov et al., 2018)	0.8339	0.8298	0.8318
TEXUS (Rastan et al., 2015)	0.8071	0.7823	0.7945

[a] Uses a domain-specific table detection model from Prasad et al., 2020.
[b] Uses a general-purpose table detection model from M. Li et al., 2020, i.e., the *X512 (ResNeXt-512) Latex+Word* model.

[18] `https://github.com/mnamysl/tabrec-sncs/tree/main/evaluation/ICDAR2013`

5.7.2 ICDAR 2019 Experiment

In this section, the results of the evaluation performed on the benchmark from ICDAR 2019 Table Competition[19] are presented.

Basic Evaluation Setup

For this experiment, the official tools and metrics used in the original competition are employed.[20] The organizers of this competition adopted the metrics employed in the ICDAR 2013 Table Competition, except that the textual content of the cells is not used for the comparison of adjacency relations, i.e., relations between the neighboring cells in a table, and the evaluation focuses on the geometrical proximity between the ground-truth and the recognized cells. The main metric used to compare the results of the examined methods is the weighted average F_1 score, abbreviated as $WAvg.F_1$, which is computed as a weighted sum of the F_1 scores obtained using four different intersection over union (IoU) thresholds for the cell matching procedure — IoU $\in \{0.6, 0.7, 0.8, 0.9\}$. IoU is defined as the ratio between the area of the overlap and the union of two bounding boxes.

Correcting an Issue in the Evaluation Script

Preliminary experiments revealed a previously unseen issue in the official evaluation script. Specifically, a simple sanity check was performed: The ground-truth data was fed as the input to the evaluation script and the expected result was to get a perfect $WAvg.F_1$ score of 1.0. Surprisingly, the obtained score of 0.793 suggested some issues with the original evaluation script. In fact, the issue in the code was located and fixed. The issue caused incorrect table matching in the case when there are two or more tables in an input image. After correcting the issue in the evaluation script, the sanity check passed as expected.

Revised Annotations

Moreover, the annotations of the track B2 (modern documents) available in the official repository hosting the data for this competition have recently been updated[21]. To estimate the expected difference in recognition scores obtained using

[19] `https://cndplab-founder.github.io/cTDaR2019`
[20] `https://github.com/cndplab-founder/ctdar_measurement_tool`
[21] `https://github.com/cndplab-founder/ICDAR2019_cTDaR/commit/66c411710a99b75a6b07f9cabce2a9480af98c78`

the revised versus the old annotations, another experiment was carried out: The old annotations were fed as input and the revised annotations were used as ground truth. Note that the corrected evaluation script was used in this experiment. The resulting $WAvg.F_1$ score of 0.647 suggests that the results obtained by evaluating against the old annotations could substantially differ from the results obtained by employing the revised annotations. This behavior could be caused by the fact that the $WAvg.F_1$ score is biased toward high overlap ratios between the cells and strongly penalizes lower IoU scores.

Final Evaluation Setup

The above-described observations motivated the author to perform the experiments in four different scenarios: (1) Using either the corrected or the original evaluation script and (2) Using either the original or the revised annotations.

In the case that the original script and the original annotations were used, the baseline methods include the best previously-reported scores on this data set.[22] In all other scenarios, ABBYY FineReader Engine[23] — a commercial solution that facilitates information extraction from documents and also provides a table recognition module — was additionally evaluated and used as a baseline approach. To adapt the output produced by this method to the format expected by the evaluation tool, the method employed in Adams et al. (2021) was used. This method parses all table blocks from the output in the ABBY-XML format and converts them to the XML format supported by the ICDAR 2019 evaluation tool.

Evaluation Results

The results of this experiment are presented in Table 5.2. In the case that the original script and the original annotations were used, the scores exhibited by the proposed method are lower than the best previously reported results in terms of $WAvg.F_1$, although the proposed hybrid system performed on par with the FineReader engine, which can be considered as a strong baseline. Interestingly, both the proposed method and the FineReader engine perform better than other methods at the highest IoU threshold. However, note that the results obtained using the evaluation script affected by the issue described in Section 5.7.2 are

[22] For fair comparison, only the prior work that employed the official tools and annotations for evaluation was included. For instance, the results of the *Multi-Type-TD-TSR* approach (Fischer et al., 2021) are omitted because the authors reannotated the test data and used images of cropped tables as input.

[23] `https://www.abbyy.com/ocr-sdk` (SDK v12).

underestimated and should be taken with a grain of salt because it is not clear how other methods would have performed if the corrected script had been used.

Moreover, the proposed method considerably outperformed ABBYY FineReader Engine in the scenarios, where the revised annotations were employed. In the remaining scenario, where the corrected script and the original annotations were used, both the proposed and the FineReader methods exhibited comparable $WAvg.F_1$ scores. Furthermore, consistent with the results presented in Section 5.7.1, the variant of the proposed method that employed the domain-specific table detection model outperformed the general-purpose variant. In summary, the best results were obtained when both the corrected evaluation script and the revised annotations were used.

Table 5.2: Evaluation results on the ICDAR 2019 benchmark (track B2 — modern document subset). The results of four different variants of the evaluation are included — using either the original or the corrected evaluation script as well as using either the original or the revised ground-truth annotations. $WAvg.F_1$ denotes the average F_1 score weighted by the IoU threshold for IoU $\in \{0.6, 0.7, 0.8, 0.9\}$.

Corrected Script	Revised Annotations	Method	$WAvg.F_1$
✗	✗	CascadeTabNet (Prasad et al., 2020)	**0.232**
		NLPR-PAL (L. Gao et al., 2019)	0.206
		Proposed[a]	0.191
		FineReader v12	0.190
		Proposed[b]	0.181
✓	✗	FineReader v12	**0.248**
		Proposed[a]	0.244
		Proposed[b]	0.220
✗	✓	Proposed[a]	**0.277**
		Proposed[b]	0.255
		FineReader v12	0.235
✓	✓	Proposed[a]	**0.345**
		Proposed[b]	0.309
		FineReader v12	0.290

[a] Uses a domain-specific table detection model from Prasad et al., 2020.
[b] Uses a general-purpose table detection model from M. Li et al., 2020, i.e., the *X512 (ResNeXt-512) Latex+Word* model.

5.8 Table Interpretation Experiment

In this section, the proposed table extraction system is examined in the end-to-end table extraction scenario. The basic variant of the proposed system is employed (Section 5.4), as it performs reliably in the scenario where the tabular layout is well defined, i.e., it follows the fully or partially bordered format, and the table labels are present.

5.8.1 Data Sets

For table interpretation, a common, publicly available benchmark could hardly be found, neither for general data nor for a specific use case. Therefore, to evaluate the proposed table interpretation method, 13 documents containing tables from an internal biomedical data collection were selected and manually annotated. Specifically, the documents containing tables presenting the inhibitory activity of different compounds toward the histone deacetylase (HDAC) gene were chosen.[24]

The ground-truth data for a table consists of a list of tuples, each representing an intersection of a data row and the columns that correspond to the defined meanings. The annotations were stored in JSON files with the following name pattern:

```
<FILE_ID>_<PAGE_NR>_<TABLE_IDX>.json
```

where `<FILE_ID>` is the file identifier, `<PAGE_NR>` is the page number in the corresponding PDF file, and `<TABLE_IDX>` is the index of a table on a page.

113 tuples from 17 tables were manually annotated and used as ground-truth data in the experimental evaluation. An example of a ground-truth file is presented in Figure 5.18. Moreover, a separate development set of four documents was selected for fine-tuning the table interpretation rules employed by the proposed approach.

5.8.2 Evaluation Setup

To perform end-to-end table extraction, the complete pipeline presented in Figure 5.2 needs to be executed. Specifically, a document in PDF format is fed as input to the system. Subsequently, preprocessing is carried out followed by table

[24] https://en.wikipedia.org/wiki/Histone_deacetylase. Specifically, the HDAC1 and HDAC6 target genes were chosen.

```
[
    {
        "compound": "9b (IC50;nM)",
        "hdac1_ic50": "84.9 \u00b1 25.1",
        "hdac6_ic50": "95.9 \u00b1 0.78"
    },
    {
        "compound": "SAHA (IC50;nM)",
        "hdac1_ic50": "102.7 \u00b1 5.9",
        "hdac6_ic50": "198.5 \u00b1 103.0"
    }
]
```

Figure 5.18: An example of a ground-truth file from the collection used in the table interpretation experiment (`11_page07_table0.json`). Note that `0x00B1` is the encoding of the '±' symbol in Unicode.

recognition and table interpretation. As a result, the relevant tuples of information are extracted from the tables contained in the input documents.

To facilitate evaluation, the extracted tuples for each table are stored in a separate JSON file (Figure 5.13c) using the same file name pattern as in the case of the ground-truth files (Section 5.8.1).

Hyperparameters and Table Interpretation Rules

The hyperparameters of the table recognition method and the rules employed by the table interpretation approach were tuned using the documents from the development set (Section 5.8.1). The following hyperparameter values of the table recognition method are used: The table labels are required by the heuristic for partially bordered tables (Section 5.4.4) and $\gamma = 2.0$ is used. Moreover, Figure A.3 in the appendix presents the final set of rules employed by the table interpretation method in the experiment presented in this section.

Evaluation Procedure

The evaluation script takes two sets of JSON files that correspond to the ground-truth and the recognized tables, respectively, as input. For every page, the script creates a bipartite graph with two sets of nodes corresponding to the ground-truth and the recognized tables, respectively. The weights of the edges in the graph correspond to the F_1 scores of the matching between the corresponding tables. Moreover, to prune all irrelevant edges, the affinity score threshold is set to $S_{min} = 0.01$.

Subsequently, maximum weight matching, as proposed by Edmonds (1965), is performed to find the correspondence between the aforementioned sets of tables. Figure 5.19 illustrates the evaluation procedure performed on a document page with two ground-truth tables.

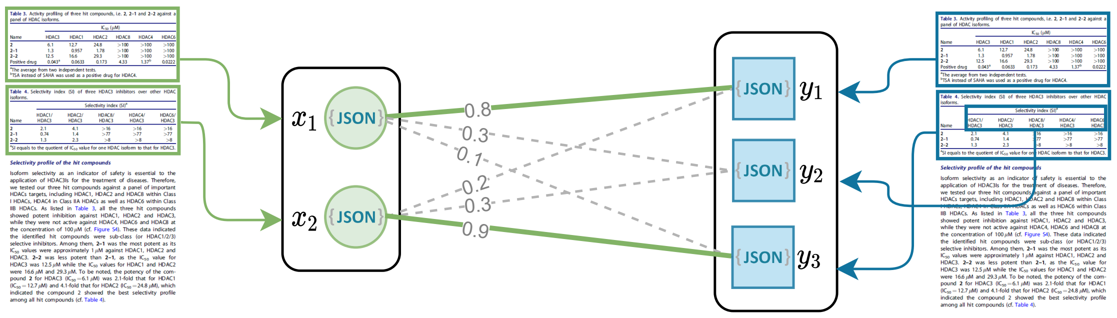


Figure 5.19: An example of a weighted bipartite interpretation graph that contains two ground-truth and three recognized tables, represented by green circles and blue squares, respectively. Each vertex in a graph corresponds to a set of tuples extracted from a table and stored in a separate JSON file. The edges are weighted by the F_1 scores of the matching between the corresponding sets of tuples. The matching with the maximum sum of weights is marked with green solid lines. Note that the y_2 vertex corresponds to a false-positive result, which is not included in the final matching. The document pages presented in this example were extracted from the employed data set.

Evaluation Metrics

To compute cumulative scores, the results from all pages are collected and the exact precision, recall, and F_1 scores are calculated. Note that the tuples from the missed reference tables and incorrectly extracted relations are also included in the reported results. Therefore, the obtained scores reflect the performance of the complete table extraction process.

It is worth noting that the evaluation script used in the experiment presented in this section was made publicly available to foster future research on IE from tables contained in unstructured documents.[25]

5.8.3 Evaluation Results

Table 5.3 presents the results of the proposed method in the complete table extraction experiment. The presented method extracted 74 tuples from 10 out of 28

[25] https://github.com/mnamysl/table-interpretation

tables present in the test set and achieved a solid F_1 score of 0.7380. Moreover, after decoupling the errors that result from the missed reference tables, the proposed method exhibited a high F_1 score of 0.9388, proving its utility.

Table 5.3: Results of IE from tabular data. The scores obtained both through the end-to-end table extraction process (*Proposed: end-to-end*) and solely from the correctly recognized tables (*Proposed: interpretation-only*) are included. The precision, recall, and F_1 score are reported. TP, FN, and FP refer to the number of tuples that were perfectly matched (*true-positive*), missed (*false-negative*), or incorrectly recognized (*false-positive*), respectively.

Method	TP	FP	FN	Precision	Recall	F_1
Proposed: end-to-end	69	4	45	0.9452	0.6053	0.7380
Proposed: interpretation-only	69	4	5	0.9452	0.9324	0.9388

Qualitative Analysis

It could be expected that lower recall of the complete table extraction system resulted from the errors made by its upstream components. Therefore, a qualitative analysis of the results was carried out to identify the main sources of the recognition errors. The analysis showed that only one false-positive and one false-negative error was directly related to the designed interpretation rules. The remaining errors were caused by TSR issues like incorrectly merged cells.

5.9 Summary

In this chapter, a flexible and modular table extraction system was presented. To infer the location and the exact structure of tables in unstructured documents, two heuristics that work with both PDF files with embedded text and image-based documents were developed (Sections 5.4.3 and 5.4.4). Moreover, in a hybrid variant of the proposed system, a deep learning-based table detection module was combined with the proposed TSR heuristics (Section 5.5). For semantic information extraction, a configurable, graph-based table interpretation method was introduced (Section 5.4.5).

Extensive experiments on challenging table recognition benchmarks showed that the proposed table recognition method is competitive with state-of-the-art approaches in the field (Section 5.7). In particular, an issue in the evaluation script used in ICDAR 2019 Table Competition was corrected and the results of the proposed and the baseline method were reported using both the original and the rectified script (Section 5.7.2). Moreover, the evaluation in the complete IE scenario, where the target information is extracted directly from raw documents, confirmed the utility of the presented holistic table extraction system (Section 5.8).

Future Work Directions

Future work could investigate different choices for the table detection module, preferably trained using a large, representative data set containing tables with various layouts, originating from different sources. Perspectively, various types of business and financial documents could also be processed, such as invoices or balance sheets.

Nevertheless, the most promising direction for future improvements is the incorporation of recent advances in the field of multimodal, pretrained models that exploit both visual and text information, such as the work presented recently by Y. Huang et al. (2022).

6 Position-Independent Evaluation of Table Recognition Systems

Preface

This chapter is adapted from Adams, Namysl, Kodamullil, Behnke, and Jacobs (2021)[1], previously published by Oxford University Press.[2]

Statement of Personal Contribution

The author of this thesis substantially contributed to all aspects of the previous publication (Adams, Namysl, Kodamullil, Behnke, and Jacobs, 2021), including the conception, design, and implementation of the proposed evaluation method, the preparation of the proposed benchmark data set, conducting the experimental evaluation, the analysis and interpretation of the experimental results, drafting the manuscript, the revision and final approval of the version to be published.

In this chapter, only the part of the previous publication (Adams, Namysl, Kodamullil, Behnke, and Jacobs, 2021) that was substantially contributed by the author of this thesis is presented. Note that Sections 6.1 and 6.8 were substantially adapted to match the part of the previous publication presented in this chapter. Moreover, Section 6.3 that describes the problem studied in this chapter is additionally presented.

Abstract

Table recognition systems are widely used to extract and structure quantitative information from the vast number of documents that are increasingly available from different open sources. While many systems already perform well on tables

[1] ©2021 Oxford University Press. Reprinted in Appendix C.5.
[2] Reproduced by permission of Oxford University Press (https://academic.oup.com).

with a simple layout, tables in the biomedical domain are often much more complex. In this chapter, a novel evaluation method of the CTR process is presented. It allows evaluating a scenario, where the exact location of table objects on a page is not available in the ground-truth annotations. The presented method was employed to evaluate several general-purpose, state-of-the-art table recognition systems on a recently released benchmark containing challenging scientific documents from the biomedical domain. Moreover, a detailed analysis of the results obtained by the baseline methods as well as a discussion of the utility of the presented approach is provided.

6.1 Introduction

Quantitative data that is present in scientific or business documents can be primarily found in tables. This data is a rich source of information that could be used to enrich the standard text mining workflows (Jimeno Yepes and Verspoor, 2014). While the number of documents and the contained quantitative information is vast, it is virtually impossible to extract this information manually due to the sheer volume of the data. Therefore, automatic table extraction systems have been widely used for this task (Section 5.3).

Although many table extraction systems already perform well on tables with a simple structure, they often struggle with tables that have a complex layout (Chi et al., 2019). Moreover, many general-purpose systems perform poorly on the examples from an out-of-domain data distribution. In particular, tables in the biomedical literature are often highly complex and therefore difficult to extract.

Furthermore, a vast number of open access, full-text biomedical publications is available both as PDF files as well as via a web browser — in an HTML-like XML format — for instance in the PubMed Central (PMC) digital repository[3]. This data has already been exploited to automatically extract the ground-truth annotations for training and evaluation of various document understanding systems, including table recognition methods (Adams and Namysl, 2021; Zhong et al., 2020). Although the structure of the tables can be extracted relatively easily from this data, determining the exact location of the table objects on a page in a PDF document is an error-prone process that is performed predominantly using heuristic-based approaches.

In this chapter, a novel evaluation method of the CTR process is presented (Section 6.4). It employs the established adjacency relation-based protocol and

[3] https://www.ncbi.nlm.nih.gov/pmc

enables performing the evaluation in a scenario, where the exact location of table objects on a page is not available in the ground-truth annotations.

The presented method was employed to evaluate several general-purpose, state-of-the-art table recognition systems (Section 6.5.2) on the benchmark released by Adams and Namysl (2021), which contains scientific publications from the biomedical domain (Section 6.6). Moreover, a thorough discussion of the results achieved by the baseline table recognition systems, the challenges based on the employed data as well as the utility of the presented approach is provided (Section 6.7).

6.2 Related Work

The most widely used table recognition benchmarks were released by the organizers of the table recognition competitions that were held at the ICDAR conference.

The data from the first ICDAR 2013 competition[4] by Göbel et al. (2013) includes born-digital business and government PDF documents. For the evaluation of table detection, *completeness* and *purity* measures were employed (Silva, 2011). However, these measures do not discriminate between minor and major errors, therefore the precision and recall scores on the subobject level were also calculated. To evaluate the accuracy of structure recognition, one-dimensional lists of *adjacency relations* (Göbel et al., 2012) between each content cell and its neighbors were compared using precision and recall scores. The adjacency relation is a tuple that contains the text of the content cell and its neighbor, the direction of the relation between both cells, and the number of blank cells in between.

The data from the second ICDAR 2019 competition[5] by L. Gao et al. (2019) includes both modern and archival documents in the image format. The modern data set contains English and Chinese documents of different types such as scientific journals, forms, financial statements, etc. The archival part exhibits a great variety of hand-drawn tables from various historical document sources such as accounting books, stock exchange lists, production censuses, etc. The IoU metric was employed to evaluate the performance of table detection. IoU measures the ratio between the area of the overlap and the union of two arbitrary shapes. The schema for table structure evaluation was adapted from the previous ICDAR competition. The annotations include table and cell positions, without the textual content of the cells.

[4] `https://roundtrippdf.com/en/data-extraction/icdar-2013-table-competition`
[5] `https://github.com/cndplab-founder/ICDAR2019_cTDaR`

Chi et al. (2019) introduced *SciTSR*[6], a large-scale TSR data set. Their benchmark was constructed by compiling the LaTeX table snippets, which were extracted from the source code crawled from *arXiv*[7] to the individual PDF files. They obtained the structure information by analyzing each table snippet to extract cell coordinates (i.e., row and column indices) and textual contents. Their method produced reference annotations in JSON format. Moreover, they divided their benchmark into *easy* and *complicated* subsets. The latter consists of tables that contain at least one cell that spans multiple rows or columns.

M. Li et al. (2020) proposed *TableBank*[8], an image-based data set of open-domain business and scientific documents in many languages, including English, Chinese, Japanese, and Arabic. They employed a weakly supervised method to build the reference annotations from Word and LaTeX documents that were crawled from the web. The authors manually added a bounding box using the markup language in the source code of each document to facilitate the automatic localization of table regions. The arrangement of rows and columns was represented as a sequence of HTML tags. They applied precision, recall, and F_1 score measures for the evaluation of table detection accuracy. For table structure assessment, the 4-gram bilingual evaluation understudy (BLEU) score (Papineni et al., 2002) with a single reference was employed.

Recently, Zhong et al. (2020) introduced *PubTabNet*[9], another image-based TSR data set that contains table images extracted from biomedical documents. Their benchmark was automatically generated by matching the table nodes in the XML files and the content of the PDF articles in PMC Open Access Subset[10]. Their algorithm produces pairs of table images with the corresponding table representation in HTML format. They employed the *tree edit distance*-based similarity metric (Pawlik and Augsten, 2016) for table structure evaluation. It is worth noting that their evaluation schema assumes that the accurate location of each table in the documents is given beforehand.

Table 6.1 summarizes the statistics of the table recognition benchmarks. Note that it also includes the benchmark released by Adams and Namysl (2021), which is described in Section 6.3. One-half of these data sets contain the documents in image format, which makes information extraction harder and less reliable. Consequently, the OCR technology needs to be additionally employed to extract the text depicted in each cell, which can lead to information loss.

[6] `https://github.com/Academic-Hammer/SciTSR`
[7] `https://arxiv.org`
[8] `https://github.com/doc-analysis/TableBank`
[9] `https://developer.ibm.com/exchanges/data/all/pubtabnet`
[10] `https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist`

Table 6.1: The number of tables in the previously proposed table detection, TSR, and CTR benchmarks. Table detection involves locating the coordinates of a table in a document. TSR takes the location of a table in a document and recognizes its structure (e.g., coordinates of the cells or HTML-like output). CTR involves both table detection and TSR. Only the data sets that provide the annotations for TSR or CTR are included.

Data set	Table Detection	TSR	CTR
ICDAR 2013	156	156	156
ICDAR 2019	1,639	750	250
SciTSR	-	15,000	-
TableBank	417,234	145,463	-
PubTabNet	-	568,592	-
Adams and Namysl (2021)	-	-	1,650

6.3 Problem Definition

Recently, a benchmark data set for evaluating table recognition systems in the biomedical literature context was released by Adams and Namysl (2021). This benchmark is based on a hand-curated literature corpus on neurological disorders and contains PDF documents with tables of varying complexities, ranging from simple $n \times m$ grid layouts to very complex structures with several nested compartments. They retrieved the PDF documents, as well as their structural representation in XML format, from PMC. Subsequently, the annotations for TSR were automatically extracted and stored in a cell-based format that was established by the ICDAR 2013 Table Competition, which specifies a flat list of cells for each table region. Moreover, in this format, table regions are structured by the start and end positions of their row and column span.

The information that is available in the original PMC-XML format allows the reconstruction of the structured content of the tables. However, the annotations for the location of the tables in a document are not available. Unfortunately, the existing protocols cannot be employed for evaluation of the CTR process in a scenario, where the reference data is only partially annotated, i.e., it does not contain the information about the location of the table objects on a page.

A straightforward solution would be to match the PDF documents with the corresponding XML files to locate the table object using manually engineered heuristics. Unfortunately, this process could be error-prone, and the results may require manual curation. Therefore, this chapter aims to answer the question: *How to evaluate the CTR process without using the information about the exact* *location of the tables in a document*?

The existing evaluation protocols either assume that the location of the table objects is known beforehand, or it is not required as the inputs are assumed to contain only one table object per document (Section 6.2). However, in a real-world scenario, tables could be annotated with the corresponding arrangement of the cells and their textual content. The HTML format is a representative example of this type of annotation. Although widely available, these data could not be fully exploited by the existing approaches.

6.4 Proposed Evaluation Method of the CTR Task

The approach presented in this section extends the portfolio of the available evaluation methods of the CTR process to the scenario, where the ground-truth data does not contain the information about the positions of the table objects on a page in an input document, thereby addressing the issue described in Section 6.3.

To this end, a novel method that evaluates the CTR process is proposed. It performs table matching and structure comparison between the ground-truth and the recognized tables using exclusively the information about the structure of the tables and not their exact locations in a document.

First, a method that measures the similarity between the relative structure of two tables was developed. Specifically, for a pair of tables (a recognized and a ground-truth table), it extracts the local adjacency relations between the cells of each table. Then, the intersection between these two sets of relations is computed and the precision, recall, and F_1 scores are calculated, as described by Göbel et al. (2012). The F_1 score calculated in this way is used to rate the similarity between two arbitrary tables and is referred to as the *affinity score*.

Note that for the computation of the adjacency relations, the same algorithm is employed as in the work of Göbel et al. (2012) except that the number of empty cells is not used, which allows computing the adjacency relations regardless of the number of empty cells between the neighboring elements. Figure 6.1 visualizes this aspect in a practical example.

To perform table matching in a document, a document-level bipartite graph with two disjoint sets of vertices, each representing the set of ground-truth and the recognized tables ($\mathcal{X}$ and $\mathcal{Y}$ in Figure 6.2, respectively) is created. Moreover, an edge is added between every pair of tables from both sets of vertices. Each edge is weighted by the affinity score between the corresponding tables. To improve performance, the connections with affinity scores equal to zero are pruned.

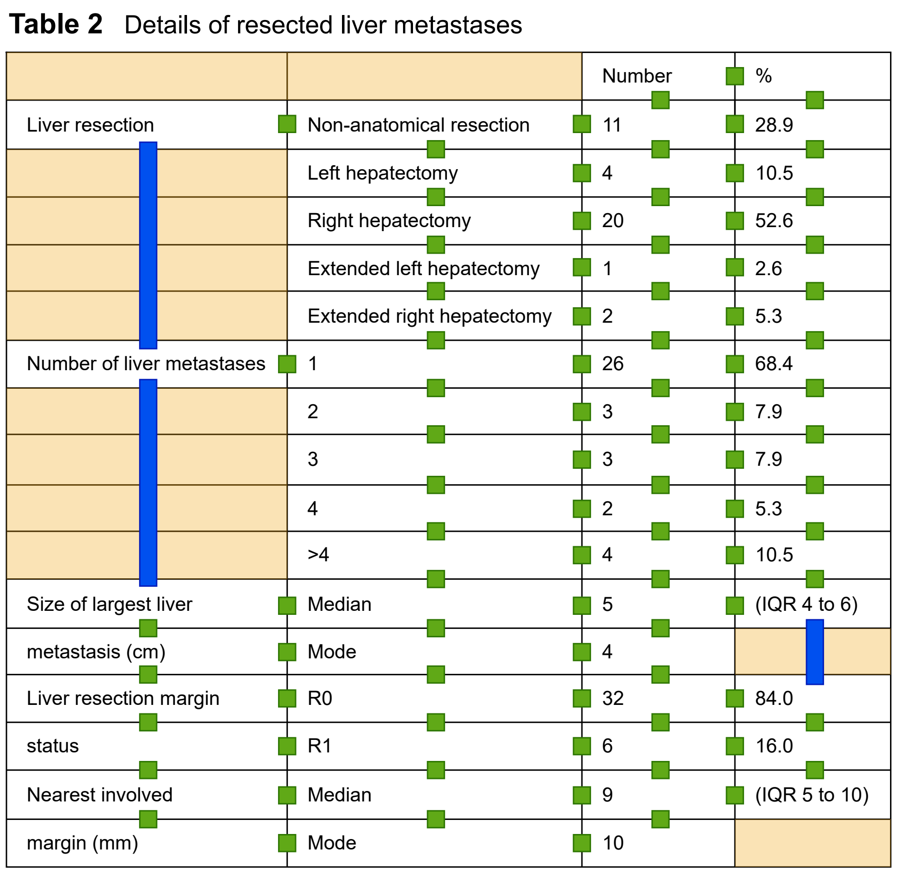


Table 2 Details of resected liver metastases

		Number	%
Liver resection	Non-anatomical resection	11	28.9
	Left hepatectomy	4	10.5
	Right hepatectomy	20	52.6
	Extended left hepatectomy	1	2.6
	Extended right hepatectomy	2	5.3
Number of liver metastases	1	26	68.4
	2	3	7.9
	3	3	7.9
	4	2	5.3
	>4	4	10.5
Size of largest liver	Median	5	(IQR 4 to 6)
metastasis (cm)	Mode	4	
Liver resection margin	R0	32	84.0
status	R1	6	16.0
Nearest involved	Median	9	(IQR 5 to 10)
margin (mm)	Mode	10	

Figure 6.1: Visualization of adjacency relations. In this example, there are 91 such relations. Green squares represent relations between the data cells. Blue rectangles indicate the relations between the cells that are separated by empty cells (marked orange). The relations will be correctly computed regardless of the number of blank cells in between. The example was adapted from Mole et al. (2011) and used under the terms of Creative Commons Attribution-NonCommercial-Share Alike 3.0 Unported License (`https://creativecommons.org/licenses/by-nc-sa/3.0`).

Subsequently, the *maximum weight matching*, as introduced by Edmonds (1965), is performed on the document-level graph to find the best correspondence between the aforementioned sets of tables. Figure 6.2 presents an example of a document-level graph with the best assignment that was found using the proposed method.

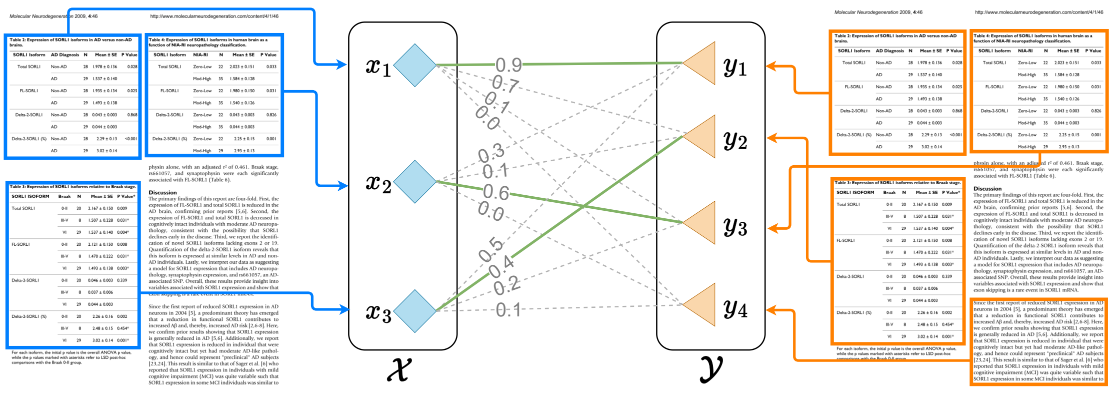


Figure 6.2: An example of a weighted document-level bipartite graph that represents three ground-truth tables (blue diamonds) and four tables recognized by a table recognition system (orange triangles) that need to be matched. The weights of the edges are calculated based on the affinity scores between the corresponding tables. Green solid lines represent the matching with the maximum sum of weights. The y_4 vertex corresponds to a false-positive result and was not included in the best matching. The document pages presented in this example were extracted from the employed benchmark data set (Adams and Namysl, 2021).

6.5 Experimental Setup

6.5.1 Data Set

In this chapter, the benchmark released by Adams and Namysl (2021) is used for the experimental evaluation of table recognition methods. This data set is comprised of 863 reference PMC documents with the corresponding annotations in XML format.

For many references in this data set, multiple PDF documents are available, as each reference corresponds to one PMC article, which, in addition to the main publication content, may contain multiple files with supplementary material. Therefore, the total number of PDF files in this data set is 1,164. The respective total number of annotated tables is 1,650.

Moreover, the tables are divided in terms of complexity, and consequently, expected difficulty for the table recognition methods, into the following three categories:

1. *Simple tables* with an $n \times m$ grid structure without any cells spanning over several rows and columns.
2. *Complicated tables* that contain a row or column (primarily found in the table headers) with cells that span over several rows or columns.
3. *Complex tables* that contain several distinct rows or columns with cells that span over other cells, which visually results in a complex, *nested* cell structure.

The respective number of tables classified into each of the above-defined categories can be found in Table 6.2. Note that even though the number of *complex tables* is rather high, the benchmark still contains a large number of *simple tables*, making the data set well balanced for testing different table recognition approaches.

Table 6.2: The number of tables in the employed benchmark data set assigned to their respective complexity category and the number of the corresponding adjacency relations.

Type	Tables	Adjacency Relations
Full benchmark	1,650	198,182
Simple tables	1,022	108,528
Complicated tables	140	20,527
Complex tables	488	69,127

Chi et al. (2019) use a similar system for the categorization of their benchmark tables. They describe tables that span over at least two columns or rows as *complicated*, similar to the categorization used in this work. They do, however, not further differentiate between tables that show this characteristic in a single row (e.g., the header) and tables that have different spanning cells in several distinct areas, which, in this work, are described as *complex*.

6.5.2 Baseline Table Recognition Methods

The following methods are experimentally evaluated on the employed benchmark data set (Section 6.5.1):

1. The basic variant of the table recognition method presented in Section 5.4. It supports the table formats that are frequently found in the scientific

literature and business documents, i.e., partially bordered tables in book tabs format[11] and fully bordered tables.

2. *ABBYY FineReader Engine*[12], a commercial solution for textual information extraction from born-digital or digitized documents. It also provides a table recognition module.
3. *TabbyPDF*[13] (Shigarov et al., 2018), an open-source tool for extracting tables from PDF documents with embedded text. It implements heuristics that leverage both textual and graphical features such as distances, fonts, and rulings.
4. *Tabula*[14], a popular open-source tool for liberating tabular data from PDF files. It implements the algorithms proposed by Nurminen (2013). Both the *lattice* and *stream mode* are employed for bordered and borderless tables, respectively. Note that this method does not support row and column spans, therefore it may underperform on the examples that contain many cells spanning multiple rows or columns.

Although many deep learning-based methods were recently proposed (Paliwal et al., 2019; Prasad et al., 2020), they predominantly work with the image-based input and are employed to recognize exclusively the structure of tables. As the goal of the experiments in this chapter is to evaluate the accuracy of the extraction of both the structure and the textual content present in the tabular data, these methods could not be employed in this scenario.

6.5.3 Evaluation Metrics

Following the previous work of Göbel et al. (2013), the precision, recall, and F_1 scores of the CTR process are reported as the main metrics in the experiments. The scores are calculated based on the adjacency relations between the cells and their neighbors (Göbel et al., 2012). Note that the adjacency relations from both false-positive recognition results and missed reference tables (false-negatives) are also included in the reported results. To calculate the cumulative scores, all *true-positive*, *false-positive*, and *false-negative* relations are first summed. Subsequently, these sums are used to calculate the final precision, recall, and F_1 scores.

[11] `https://ctan.org/pkg/booktabs`
[12] `https://www.abbyy.com/ocr-sdk` (SDK v12).
[13] `https://github.com/cellsrg/tabbypdf` (Retrieved: 10/20/2020).
[14] `https://github.com/tabulapdf/tabula-java` (v1.0.4).

6.5.4 Evaluation Procedure

To evaluate the baseline table recognition methods (Section 6.5.2) on the employed benchmark data set (Section 6.5.1), the algorithm described in Section 6.4 was used. To fully exploit the data, further practical extensions to this procedure were additionally implemented.

Unicode Normalization

To mitigate the negative impact of negligible differences between the ground-truth annotations from the XML files and the text extracted from the input PDF files, *Unicode normalization*[15] is performed on the textual content of both the ground-truth and the recognized cells. Specifically, all ligatures are decomposed into their components, e.g., the ligature 'fi' (`0xFB01`) is decomposed to '`f`' and '`i`'.

Multiple-Variant Evaluation

Note that, for some articles, multiple PDF files are available — the main article and its appendices. The reference XML file content may correspond to any of these PDF files. Therefore, the matching described in Section 6.4 is first performed with the recognition results obtained for all PDF files available for a particular article. Subsequently, the document with the highest F_1 score is selected to be included in the final results.

In the remainder of this chapter, the basic evaluation procedure that uses only the first PDF document, which is available for an article, and the extension described in this paragraph are referred to as the *single-variant* and *multiple-variant* evaluation, respectively.

Subset-Level Evaluation

To analyze how the examined table recognition systems handle table layouts with different complexities (Table 6.2), the subset-level evaluation was additionally performed. In this scenario, the single-variant procedure is employed to enforce consistent matching across all methods. Moreover, the relations from all false-positively detected tables, i.e., recognized tables that were not matched with any ground-truth table (see y_4 in Figure 6.2) are reported in a separate subset.

[15] http://unicode.org/reports/tr15

6.6 Experimental Results

6.6.1 Single-Variant and Multiple-Variant Evaluation Results

The results of the single-variant and the multiple-variant evaluation (Section 6.5.4) are presented in Table 6.3. In terms of the F_1 score metric, ABBYY FineReader Engine outperformed all other methods, followed by the approach presented in Section 5.4 and the TabbyPDF system. Moreover, the method presented in Section 5.4 exhibited the best precision among all competitors. Furthermore, Tabula was outperformed by other competitors by a large margin. Neither the stream nor the lattice mode could reliably handle tables present in the employed benchmark data set.

As expected, the multiple-variant evaluation procedure improved recall and the number of correctly recognized relations for all examined table recognition methods. Moreover, the multiple-variant evaluation also improved the F_1 scores (except for Tabula in the lattice mode).

Summarizing the results of the best methods, there is still plenty of room for improvements and the employed benchmark proved to be very challenging even for the current state-of-the-art table recognition systems.

6.6.2 Subset-Level Evaluation Results

The results of the subset-level evaluation are presented in Table 6.4. The analysis is limited to the top three methods from the experiment in Section 6.6.1. Note that the relations from all false-positively detected tables are reported separately (see *False Positives* in Table 6.4).

ABBYY FineReader Engine outperformed all other baselines in terms of the F_1 score on all three table complexity subsets. Moreover, the method presented in Section 5.4 recognized the smallest number of false-positive relations in all evaluation scenarios and exhibited the highest precision except for the evaluation on the subset of *complex* tables.

In general, the employed benchmark data set is of high difficulty. In particular, the *complex* subset containing nested tables was the most challenging for all methods examined in this experiment. Apparently, the achieved results can most likely be improved, considering that the methods were not tuned for this particular data set.

Table 6.3: Evaluation results on the full benchmark using the single-variant and the multiple-variant method (Section 6.5.4). The Precision (*Prec.*), recall, and F_1 scores are reported for the CTR process. Moreover, the number of the found, correctly recognized (TP, *true-positive*), missed (FN, *false-negative*), and incorrectly recognized (FP, *false-positive*) relations are presented. The results are sorted by F_1 scores. **Bold** values indicate the best results (highest scores, the lowest number of incorrect relations, etc.).

(a) Single-variant method

Method	Relations				Prec.	Recall	F_1
	Found	TP	FN	FP			
ABBYY FineReader	196,807	**117,904**	**80,278**	78,903	0.5991	**0.5949**	**0.5970**
This work (Section 5.4)	138,835	92,666	105,516	46,169	**0.6675**	0.4676	0.5499
TabbyPDF	215,540	108,207	89,975	107,333	0.5020	0.5460	0.5231
Tabula (lattice mode)	40,754	25,689	172,493	**15,065**	0.6303	0.1296	0.2150
Tabula (stream mode)	693,723	69,435	128,747	624,288	0.1001	0.3504	0.1557

(b) Multiple-variant method

Method	Relations				Prec.	Recall	F_1
	Found	TP	FN	FP			
ABBYY FineReader	199,103	**120,661**	**77,521**	78,442	0.6060	**0.6088**	**0.6074**
This work (Section 5.4)	144,228	95,199	102,983	49,029	**0.6601**	0.4804	0.5561
TabbyPDF	206,576	111,195	86,987	95,381	0.5383	0.5611	0.5494
Tabula (lattice mode)	47,699	26,086	172,096	**21,613**	0.5469	0.1316	0.2122
Tabula (stream mode)	681,872	71,693	126,489	610,179	0.1051	0.3618	0.1629

Table 6.4: Evaluation results at the subset level (Section 6.5.4). The Precision (*Prec.*), recall, and F_1 scores for the CTR process are reported. Moreover, the number of the found, correctly recognized (TP, *true-positive*), missed (FN, *false-negative*), and incorrectly recognized (FP, *false-positive*) relations are presented. Results are sorted by F_1 scores. **Bold** values indicate the best results (highest scores, the lowest number of incorrect relations, etc.) within each subset.

(a) Simple Tables

Method	Relations				Prec.	Recall	F_1
	Found	TP	FN	FP			
ABBYY FineReader	86,909	**68,177**	**40,351**	18,732	0.7845	**0.6282**	**0.6977**
This work (Section 5.4)	72,651	57,293	51,235	**15,358**	**0.7886**	0.5279	0.6324
TabbyPDF	90,894	62,592	45,936	28,302	0.6886	0.5767	0.6277

(b) Complicated Tables

Method	Relations				Prec.	Recall	F_1
	Found	TP	FN	FP			
ABBYY FineReader	17,496	**14,046**	6,481	3,450	0.8028	**0.6843**	**0.7388**
TabbyPDF	12,728	7,799	**2,675**	44,376	0.8263	0.6201	0.7085
This work (Section 5.4)	11,217	9,274	11,253	**1,943**	**0.8268**	0.4518	0.5843

(c) Complex Tables

Method	Relations				Prec.	Recall	F_1
	Found	TP	FN	FP			
ABBYY FineReader	49,627	**36,074**	**35,681**	13,946	**0.7190**	**0.5162**	**0.6009**
TabbyPDF	52,213	32,887	36,240	19,326	0.6299	0.4757	0.5421
This work (Section 5.4)	39,413	26,099	43,028	**13,314**	0.6622	0.3776	0.4809

(d) False Positives

Method	Relations				Prec.	Recall	F_1
	Found	TP	FN	FP			
This work (Section 5.4)	15,554	—	—	**15,554**	—	—	—
ABBYY FineReader	42,775	—	—	42,775	—	—	—
TabbyPDF	57,030	—	—	57,030	—	—	—

6.7 Discussion

6.7.1 Table Extraction Challenges Based on Benchmark Data

In PDF files, the character shapes (*glyphs*) and their mappings to the Unicode characters (*toUnicode* lookup table) are stored separately. The former is used to render a page, i.e., to draw each glyph. The latter contains the textual representation of the glyphs, which is extracted by the PDF parsing software.

While analyzing the results of the baseline systems, a problem with corrupted or incomplete Unicode mappings in some files was identified. In such a case, the PDF content is rendered correctly, but the extracted text does not match the rendered glyphs (e.g., '$\leftrightarrow$' and '$\downarrow$' are extracted as '$' and '#', respectively). This problem could accidentally be caused by the software used to create or export PDF files.

Unfortunately, this problem cannot be automatically repaired. A possible solution would be to render each page, apply OCR, and compare the results with the extracted text. However, as a perfect OCR software still does not exist (see Section 2.1), we should also differentiate between the OCR errors and the broken *toUnicode* mappings, which would further complicate the extraction process.

6.7.2 PDF vs. Image-Based Input Documents

It is worth noting that the evaluation method presented in Section 6.4 does not depend on the format of the input documents. Although the employed benchmark consists of PDF files with embedded text, the presented method is not limited to this input format, as it uses the structure of the tables from ground-truth annotations and recognition results to calculate the final scores (precision, recall, and F_1). Note that it is up to the table recognition method to support the inputs in a particular document format (PDF, image, etc.).

6.7.3 Could We Also Evaluate the Accuracy of Table Detection?

As discussed in Section 6.4, the presented method evaluates the CTR process using exclusively the information about the content and the structure of the tables. It does not rely on the positions and the boundaries of table objects, because this information is not present in the ground-truth XML files contained in the employed benchmark (Section 6.5.1).

Nevertheless, the same method could also be indirectly used to evaluate the table detection task by using the affinity score as a proxy for the IoU metric, which is commonly used to rate the performance of table detection systems. Consequently, all matchings (edges in the bipartite document graph in Figure 6.2) with affinity scores equal to or greater than a predefined threshold could be considered as correctly detected. The standard metrics such as precision, recall, and F_1 score might be then used to evaluate table detection accuracy. Such an evaluation could be thus regarded as a variant of the presented method with additional constraints, i.e., the affinity score threshold.

6.7.4 Evaluation Behavior on Two Practical Examples

In this section, the behavior of the presented evaluation method is analyzed on two practical examples illustrated in Figure 6.3.

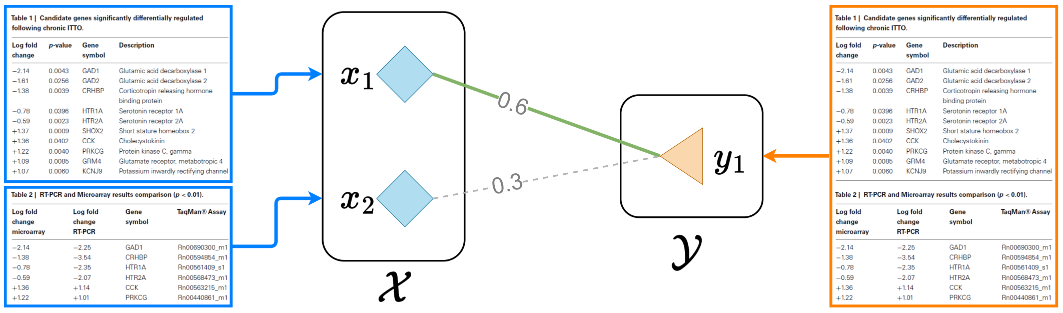


Table 1 | Candidate genes significantly differentially regulated following chronic ITTO.

Log fold change	*p*-value	Gene symbol	Description
−2.14	0.0043	GAD1	Glutamic acid decarboxylase 1
−1.61	0.0256	GAD2	Glutamic acid decarboxylase 2
−1.38	0.0039	CRHBP	Corticotropin releasing hormone binding protein
−0.78	0.0396	HTR1A	Serotonin receptor 1A
−0.59	0.0023	HTR2A	Serotonin receptor 2A
+1.37	0.0009	SHOX2	Short stature homeobox 2
+1.36	0.0402	CCK	Cholecystokinin
+1.22	0.0040	PRKCG	Protein kinase C, gamma
+1.09	0.0085	GRM4	Glutamate receptor, metabotropic 4
+1.07	0.0060	KCNJ9	Potassium inwardly rectifying channel

Table 2 | RT-PCR and Microarray results comparison ($p < 0.01$).

Log fold change microarray	Log fold change RT-PCR	Gene symbol	TaqMan® Assay
−2.14	−2.25	GAD1	Rn00690300_m1
−1.38	−3.54	CRHBP	Rn00594854_m1
−0.78	−2.35	HTR1A	Rn00561409_s1
−0.59	−2.07	HTR2A	Rn00568473_m1
+1.36	+1.14	CCK	Rn00563215_m1
+1.22	+1.01	PRKCG	Rn00440861_m1

(a) Illustration of the first example (a detected table covers two ground-truth tables)

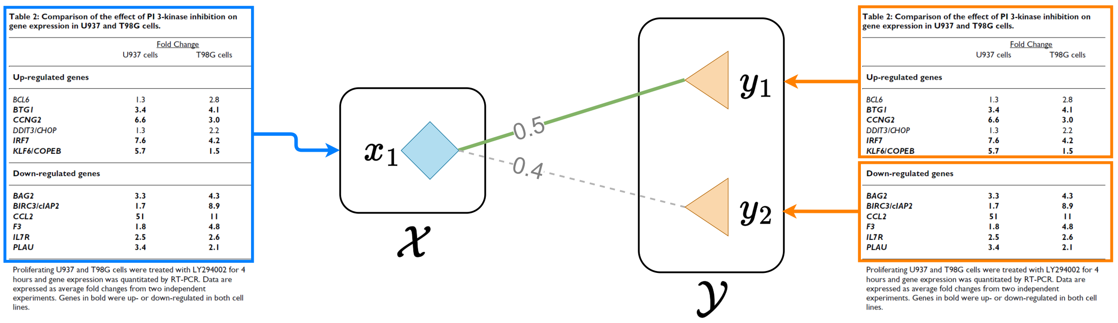


Table 2: Comparison of the effect of PI 3-kinase inhibition on gene expression in U937 and T98G cells.

	Fold Change	
	U937 cells	T98G cells
Up-regulated genes		
BCL6	1.3	2.8
BTG1	**3.4**	**4.1**
CCNG2	**6.6**	**3.0**
DDIT3/CHOP	1.3	2.2
IRF7	**7.6**	**4.2**
KLF6/COPEB	**5.7**	**1.5**
Down-regulated genes		
BAG2	**3.3**	**4.3**
BIRC3/cIAP2	**1.7**	**8.9**
CCL2	**51**	**11**
F3	**1.8**	**4.8**
IL7R	**2.5**	**2.6**
PLAU	**3.4**	**2.1**

Proliferating U937 and T98G cells were treated with LY294002 for 4 hours and gene expression was quantitated by RT-PCR. Data are expressed as average fold changes from two independent experiments. Genes in bold were up- or down-regulated in both cell lines.

(b) Illustration of the second example (a ground-truth table erroneously detected as two tables)

Figure 6.3: Evaluation behavior demonstrated using two practical examples. Blue diamonds and orange triangles represent the ground-truth and the recognized tables, respectively. Green solid lines depict the best matching. The x_2 and y_2 vertices correspond to a false-negative and false-positive detection result, respectively. The document pages presented in this example were extracted from the employed benchmark data set (Adams and Namysl, 2021).

Example 1 The ground-truth data contains two tables, but the recognition algorithm detects one table that covers both ground-truth tables.

Example 2 The ground truth data contains one table, but two tables are erroneously detected (each of them overlaps the ground truth table).

Ad. Example 1 We can expect that the detected table will be matched with the ground-truth table that has a larger affinity score with the recognition result. In practice, it would be the table that contains more data cells. The other ground-truth table will be considered a false-negative detection.

Ad. Example 2 We can expect that the ground-truth table will be matched with the detected table that overlaps the ground truth table the most. The other detected table will be considered a false-positive detection.

6.8 Summary

In this chapter, a novel evaluation method of the CTR process was presented. It allows the evaluation of a scenario, where the information about the location of the table objects on a page is not available in the ground-truth annotations (Section 6.4).

The presented approach was used to evaluate the accuracy of several state-of-the-art, general-purpose table recognition systems on a recently proposed benchmark containing documents from a biomedical domain (Section 6.5.1). In particular, the accuracy of the baseline systems on different subsets of the data, which were categorized based on the complexity of the table layout, was reported. A thorough discussion of the results achieved by the baseline table recognition systems (Section 6.6), as well as the challenges based on the employed data and the utility of the presented evaluation method were provided (Section 6.7).

The presented graph-based table matching algorithm (Section 6.4) is universal and is not limited to the adjacency relation-based evaluation. The adaptation to other metrics is straightforward. For instance, the affinity score employed by the presented approach could also be defined using the tree edit distance-based similarity metric (Pawlik and Augsten, 2016) or the BLEU score (Papineni et al., 2002) and used to evaluate the CTR process.

Future Work Directions

Future work could explore alternative choices for the affinity score metric, as discussed in the previous paragraph. Moreover, a more sophisticated edge pruning strategy would improve the computational efficiency of the presented algorithm in a scenario, where large sets of tables need to be matched with each other. Furthermore, it would be beneficial to test the utility of the presented method using other data sets, such as TableBank (M. Li et al., 2020), and to compare the results of a standard evaluation process, which uses the information about the exact location of the tables on a page to match the recognized and the reference objects, with the results obtained using the proposed content-based, position-independent evaluation algorithm.

7 Conclusions

In this thesis, we studied the robustness of the IE systems that work with either born-digital or digitized documents. We analyzed different components of the IE workflow such as the OCR engine, the sequence labeling models, and the table extraction module. In particular, several novel techniques and enhancements that lead to improved robustness of the IE process were presented.

Firstly, an efficient and robust, deep learning-based OCR engine, which can be trained almost exclusively using synthetically generated documents, was presented. Moreover, a novel data augmentation technique that blends rendered text with a background texture image was introduced. The presented approach outperformed state-of-the-art commercial and open-source OCR engines in terms of accuracy and computational efficiency on a challenging data set that consists of both historical and contemporary documents.

Secondly, the NAT method was introduced and used to improve the robustness of a state-of-the-art sequence labeling model in a scenario, where misrecognized or mistyped text is used as input. NAT exploits both the original text and its corrupted version, which is generated by inducing errors into the original sentence using a noise model that imitates naturally occurring errors. The NAT approach was shown to mitigate the negative impact of erroneous textual input without diminishing the accuracy of the model on original, error-free sentences.

Thirdly, an extension of the NAT method was proposed, which replaces the vanilla error model used at training time with an empirical error generator that induces more plausible errors into the original text. Moreover, an unsupervised parallel training data generation method was introduced and employed to synthesize the data used to estimate the empirical error model. Furthermore, the NLM embeddings were trained using the noisy part of the generated parallel data set and integrated into the NAT framework. The proposed improvements were shown to significantly improve the robustness of the baseline NAT approach.

Fourthly, for the restoration of structural information from documents, a flexible table extraction system was presented. It consists of configurable table recognition and semantic interpretation modules that can be used to extract the desired information directly from tables contained in the input document. The presented approach achieved table recognition scores on par with the state-of-the-art methods. Moreover, the proposed holistic table extraction system exhibited high recognition accuracy in an end-to-end IE scenario, where raw documents were used as input and the target information was contained in tables.

Fifthly, a novel evaluation method of the CTR process was introduced. It enables performing the evaluation in a scenario, where the exact location of table objects on a page is not available in the ground-truth annotations, for instance, in the case of the HTML format, which is often used in digital repositories that archive open access scholarly articles. The proposed method was used to evaluate several general-purpose, state-of-the-art table recognition systems on the benchmark that contains scientific publications from the biomedical domain. In addition to the results of the experimental evaluation, a thorough discussion of the results achieved by the baseline table recognition systems in an out-of-domain scenario, the challenges based on the employed data set as well as the utility of the presented approach were provided.

7.1 Future Work Directions

The methods presented in this thesis open several opportunities for future research. In this section, the major directions are outlined. Moreover, in the summary section of each chapter, a detailed discussion of the future work directions for the corresponding approach is provided.

For instance, the presented text recognition method could be coupled with an automatic image rectification module to improve robustness to geometrical irregularities that are common in the STR scenario. Moreover, the incorporation of recent advances in the field of computer vision into the presented network architecture should provide further improvements in terms of both recognition accuracy and robustness.

The NAT approach could be combined with techniques that allow for avoiding erroneous tokenization, which would additionally help mitigate the impact of token merging and splitting errors. The NAT approach should also improve the robustness of other NLP tasks beyond the sequence labeling scenario investigated in this thesis. Moreover, the research community would further benefit from a pre-trained multilingual NLM text representation, as well as from a broader spectrum of embeddings specialized in a particular language or error distribution, which could be used to improve the robustness of various language understanding tasks.

Therefore, the incorporation of the NAT approach and the NLM embeddings into every framework that processes noisy input text is strongly advocated.

Future work on the presented table recognition approach could investigate different choices for the table detection module, preferably trained using a large, representative data set containing tables with various layouts, originating from different sources. Moreover, a promising direction for future improvements is the incorporation of recent advances in the field of multimodal, pretrained models that exploit both visual and text information.

Regarding the proposed evaluation method of the CTR process, alternative choices of the employed affinity score metric could be investigated in future work. Moreover, a comparison of the results obtained by the proposed content-based, position-independent CTR evaluation algorithm with a method that directly utilizes the location of table objects would give further insight into the utility of the presented approach.

A Supplementary Material

Table A.1: Detailed architecture of the hybrid CNN-BLSTM text recognition model (Section 2.4.1). Two blocks of two-dimensional convolutional layers (Conv2D), interleaved with the max pooling, and the batch normalization operations (Ioffe and Szegedy, 2015), are followed by the map-to-sequence module, which transforms the activation maps into feature sequences. The values in parentheses correspond to the kernel size, the number of filters, and the stride size, respectively. Subsequently, the feature sequences are fed into a BLSTM network. Dropout (Srivastava et al., 2014) is applied to the output feature vectors with a probability of 50%. The outputs from the forward and backward LSTM layers are concatenated and fed to a linear layer that produces a per timestep probability distribution over the set of available classes. Finally, either a CTC loss or a CTC decoding function is employed during training and inference, respectively. In the second column, the size of the output volume is presented, where W is the width of an input image, C is the number of classes (characters) recognized by the model, and L is the length of the output sequence of natural language tokens.

Operation	Output volume size
Conv2D ($3\times3, 64$; stride: 1×1)	$32\times W\times64$
Max pooling (2×2; stride: 2×2)	$16\times W/2\times64$
Batch normalization	—
Conv2D ($3\times3, 128$; stride: 1×1)	$16\times W/2\times128$
Max pooling (2×2; stride: 2×2)	$8\times W/4\times128$
Batch normalization	—
Map-to-sequence function	$W/4\times1024$
BLSTM (num. units: 2×256)	$W/4\times512$
Dropout (50% probability)	—
Linear mapping (num. units: C)	$W/4\times C$
CTC output layer	L

Table A.2: Detailed architecture of the FCN-based text recognition model (Section 2.4.1). It consists of ten blocks of two-dimensional convolutional layers (Conv2D), interleaved with the batch normalization operation (Ioffe and Szegedy, 2015), that gradually reduce the height of the feature maps. The values in parentheses correspond to the kernel size, the number of filters, and the stride size, respectively. Note that the first Conv2D layer is additionally followed by the max pooling operation. Subsequently, the map-to-sequence operation transforms the activation maps into feature sequences that are fed to a linear layer, which produces a per timestep probability distribution over the set of available classes. Finally, either a CTC loss or a CTC decoding function is employed during training and inference, respectively. In the second column, the size of the output volume is presented, where W is the width of an input image, C is the number of classes (characters) recognized by the model, and L is the length of the output sequence of natural language tokens.

Operation	Output volume size
Conv2D ($7{\times}7, 64$; stride: $2{\times}2$)	$16{\times}{}^{W}/_{2}{\times}64$
Max pooling ($2{\times}2$; stride: $2{\times}2$)	$8{\times}{}^{W}/_{4}{\times}64$
Batch normalization	—
Conv2D ($3{\times}3, 64$; stride: $1{\times}1$)	$8{\times}{}^{W}/_{4}{\times}64$
Batch normalization	—
Conv2D ($3{\times}3, 64$, stride: $1{\times}1$)	$8{\times}{}^{W}/_{4}{\times}64$
Batch normalization	—
Conv2D ($3{\times}3, 128$, stride: $2{\times}1$)	$4{\times}{}^{W}/_{4}{\times}128$
Batch normalization	—
Conv2D ($3{\times}3, 128$, stride: $1{\times}1$)	$4{\times}{}^{W}/_{4}{\times}128$
Batch normalization	—
Conv2D ($3{\times}3, 256$, stride: $2{\times}1$)	$2{\times}{}^{W}/_{4}{\times}256$
Batch normalization	—
Conv2D ($3{\times}3, 256$, stride: $1{\times}1$)	$2{\times}{}^{W}/_{4}{\times}256$
Batch normalization	—
Conv2D ($3{\times}3, 512$, stride: $2{\times}1$)	$1{\times}{}^{W}/_{4}{\times}512$
Batch normalization	—
Conv2D ($3{\times}3, 512$, stride: $1{\times}1$)	$1{\times}{}^{W}/_{4}{\times}512$
Batch normalization	—
Conv2D ($3{\times}3, 512$, stride: $1{\times}1$)	$1{\times}{}^{W}/_{4}{\times}512$
Batch normalization	—
Map-to-sequence function	${}^{W}/_{4}{\times}512$
Linear layer (num. units: C)	${}^{W}/_{4}{\times}C$
CTC output layer	L

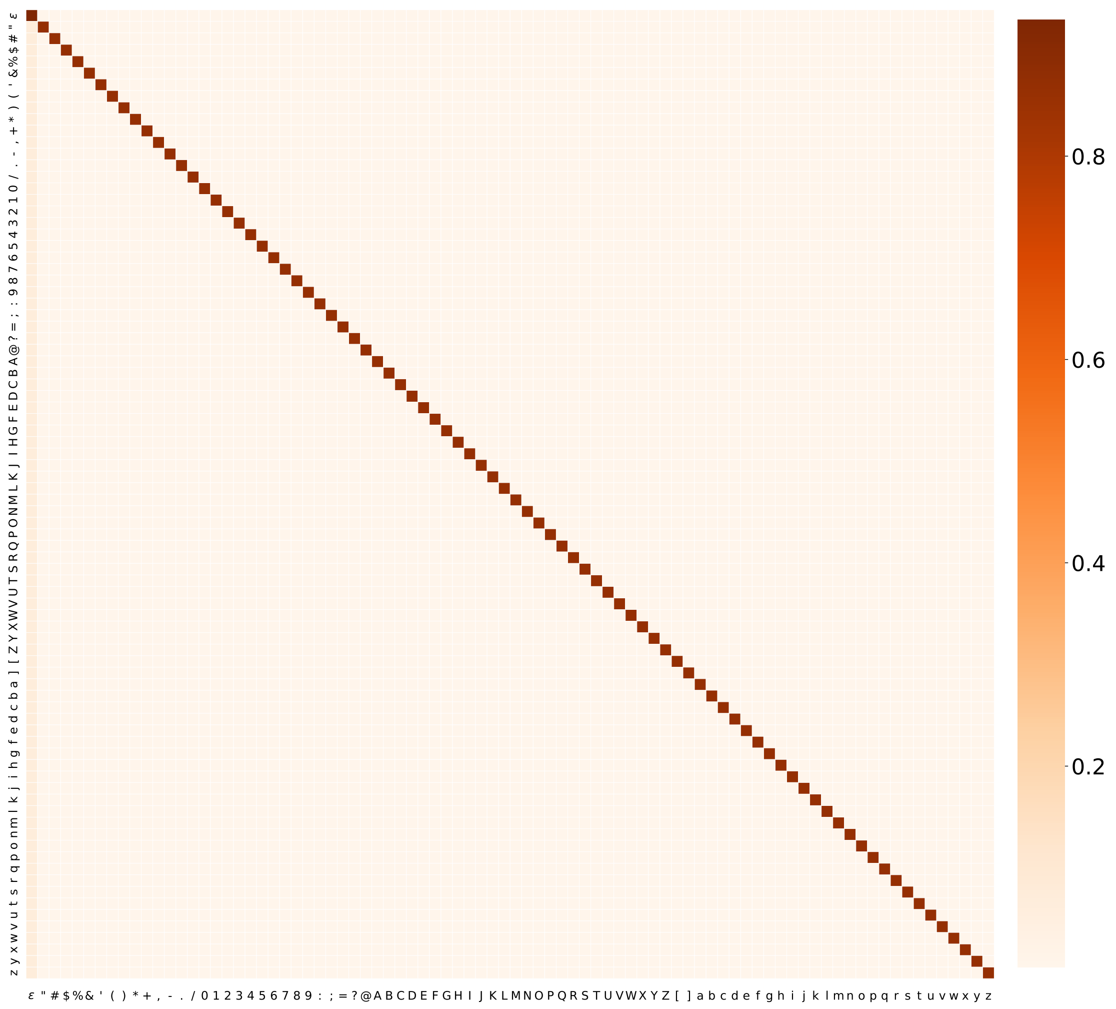


Figure A.1: Confusion matrix for the vanilla error distribution (Section 3.3.1) used at training time ($\eta = 20\%$). Each cell represents $P(\tilde{c}|c)$. The rows and the columns correspond to the original characters c and the perturbed characters $\tilde{c}$, respectively. In this example, all symbols from the alphabet of the English CoNLL 2003 data set are included. The vanilla noise model assigns equal probability to all substitution errors and assumes that the deletion of a character c is as likely as the sum of substitution probabilities with all nonempty symbols, i.e.: $P_{del}(\varepsilon|c) = \sum_{\tilde{c} \in \Sigma \setminus \{\varepsilon\}} P_{subst}(\tilde{c}|c)$.

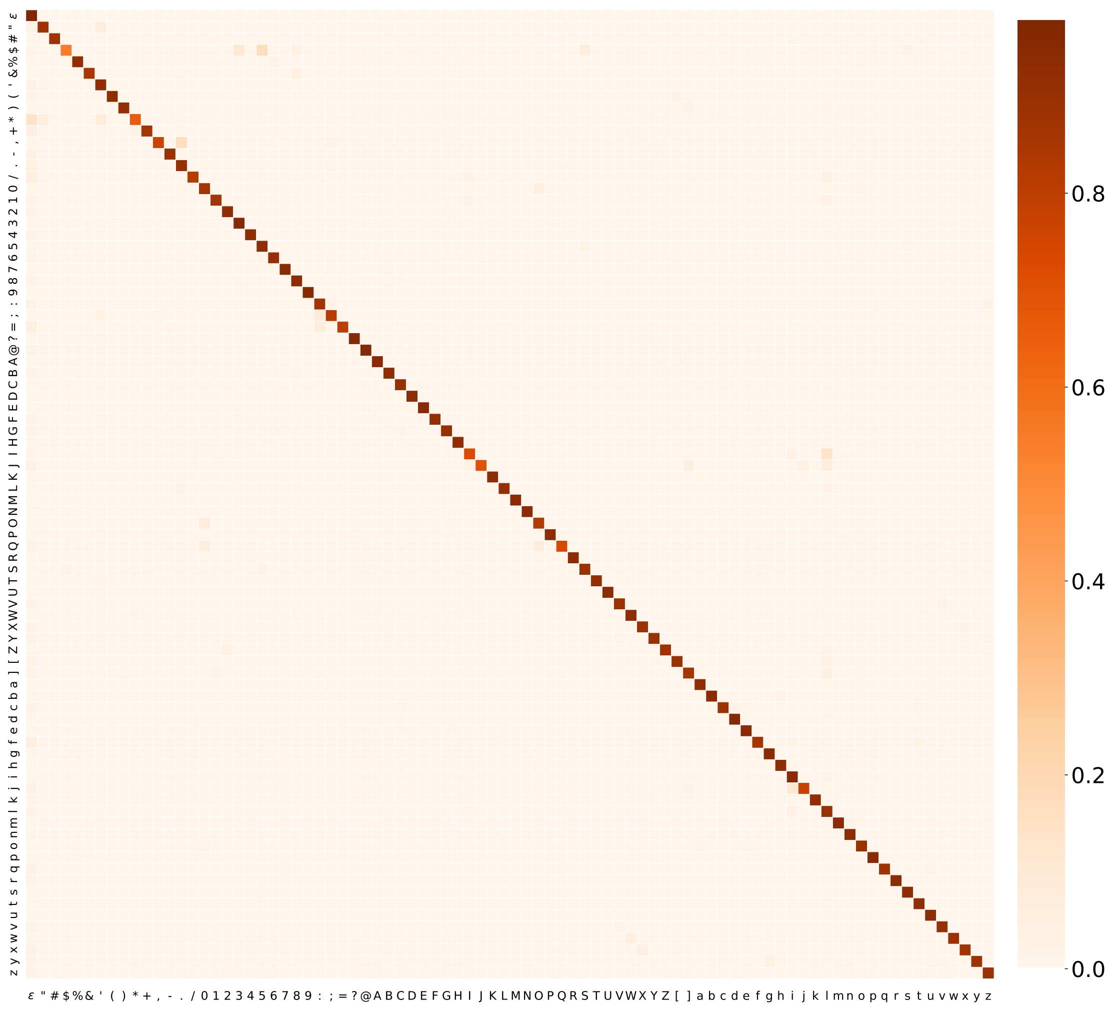


Figure A.2: Confusion matrix for the OCR error distribution estimated from a large document corpus using the Tesseract OCR engine (Section 3.3.1). Each cell represents $P(\tilde{c}|c)$. The rows and the columns correspond to the original characters c and the perturbed characters $\tilde{c}$, respectively. In this example, all symbols from the alphabet of the English CoNLL 2003 data set were included. Note that the OCR error model is biased toward substitutions of characters with similar shapes like 'I'$\to$'l', '$'$\to$'5', 'O'$\to$'0' or ','$\to$'.'.

Table A.3: Hyperparameters of the sequence labeling model $f(x)$ (Section 3.4.3). Note that dropout is applied both before and after the BLSTM layer.

Parameter name	Parameter value
Mini batch size	32
Maximum number of epochs	100
Number of BLSTM hidden layers	1
Number of BLSTM hidden units	256
Optimizer	SGD
Initial learning rate	0.1
Learning rate anneal factor	0.5
Minimum learning rate	0.0001
Word dropout level	0.05
Variational dropout level	0.5
Patience	3

Table A.4: Hyperparameters of the OpenNMT toolkit used to train the sequence-to-sequence models employed in this work (Section 4.5.1).

Parameter	Description	Value
`-share_vocab`	Share source and target vocabulary	True
`-share_embeddings`	Share the embeddings between encoder and decoder	True
`-word_vec_size`	Word embedding size	25
`-src_seq_length`	Maximum source sequence length	1,000
`-tgt_seq_length`	Maximum target sequence length	1,000
`-encoder_type`	Type of encoder layer	BRNN
`-learning_rate`	Starting learning rate	1.0

Table A.5: Hyperparameters of the TSR method used by the hybrid variant of the proposed table extraction system (Section 5.6.3). The values used both in the experiment on the ICDAR 2013 (Section 5.7.1) and the ICDAR 2019 (Section 5.7.2) benchmark are presented. Note that, in the case of ICDAR 2019, table candidates are not discarded based on H_{ratio}.

Name	ICDAR 2013	ICDAR 2019
Min. number of rows	2	2
Min. number of columns	2	2
Min. number of cells	7	7
H_{ratio}^{max} threshold	10.0	not used
Column gap threshold (γ)	1.5	1.0

```
[
  { "id": "compound",
    "keywords": ["Compound", "compd", "Comp.", "cpd"],
    "datatype": "string",
    "weightTitle": 1.0,
    "weightContent": 0.0,
    "minAffinityScore": 0.5
  },
  { "id": "hdac1_gene",
    "keywords": ["HDAC1"],
    "titleRegex": "^HDAC[-]{0,1}1[^\\d].*$",
    "datatype": ["double", "range", "integer"],
    "weightTitle": 1.0,
    "weightContent": 0.0,
    "minAffinityScore": 0.85
  },
  { "id": "hdac6_gene",
    "keywords": ["HDAC6"],
    "titleRegex": "^HDAC[-]{0,1}6[^\\d].*$",
    "datatype": ["double", "range", "integer"],
    "weightTitle": 1.0,
    "weightContent": 0.0,
    "minAffinityScore": 0.85
  }
]
```

Figure A.3: A JSON file defining the meanings and rules for matching columns to these meanings used in the table interpretation experiment (Section 5.8.2).

B Statistics of the Sequence Labeling Data Sets

In this chapter, the detailed statistics of the sequence labeling data sets employed in Chapters 3 and 4 are presented.

Table B.1: Statistics of the CoNLL 2003 data set (Sections 3.4.1 and 4.5.5). The training, development (Devel.), and test sets were included, as well as the number of sentences, tokens, and entities: person names, locations, organizations, and miscellaneous.

(a) English CoNLL 2003.

Element	Training	Devel.	Test	Total
Sentences	14,041	3,250	3,453	20,744
Tokens	203,621	51,362	46,435	301,418
Person names (PER)	6,600	1,842	1,617	10,059
Locations (LOC)	7,140	1,837	1,668	10,645
Organizations (ORG)	6,321	1,341	1,661	9,323
Miscellaneous (MISC)	3,438	922	702	5,062

(b) German CoNLL 2003.

Element	Training	Devel.	Test	Total
Sentences	12,705	3,068	3,160	18,933
Tokens	207,484	51,645	52,098	311,227
Person names (PER)	2,801	1,409	1,210	5,420
Locations (LOC)	4,273	1,216	1,051	6,540
Organizations (ORG)	2,154	1,090	584	3,828
Miscellaneous (MISC)	780	216	206	1,202

Table B.2: Statistics of the GermEval 2014 data set (Section 3.4.1). The training, development (Devel.), and test sets were included, as well as the number of sentences, tokens, and entities: person names (PER), locations (LOC), organizations (ORG), and miscellaneous (MISC). This data set defines two additional fine-grained subtags: *-part* and *-deriv* that mark derivation and compound words, respectively, which stand in direct relation to named entities.

Element	Training	Devel.	Test	Total
Sentences	24,000	2,200	5,100	31,300
Tokens	452,853	41,653	96,499	591,005
PER	7,679	711	1,639	10,029
PER-deriv	62	2	11	75
PER-part	184	18	44	246
LOC	8,281	763	1,706	10,750
LOC-deriv	2,808	235	561	3,604
LOC-part	513	52	109	674
ORG	5,255	496	1,150	6,901
ORG-deriv	41	3	8	52
ORG-part	805	91	172	1,068
MISC	3,024	269	697	3,990
MISC-deriv	236	16	39	291
MISC-part	190	18	42	250

Table B.3: Statistics of the CoNLL 2000 data set (Section 3.4.1). The training and test sets were included, as well as the number of sentences and tokens. This data set has no development part. Note that the NP, VP, and PP chunks account for over 90% of all chunk types.

Element	Training	Test	Total
Sentences	8,936	2,012	10,948
Tokens	211,727	47,377	259,104
Noun phrase (NP)	55,081	12,422	67,503
Verb phrase (VP)	21,467	4,658	26,125
Prepositional phrase (PP)	21,281	4,811	26,092
Adverb phrase (ADVP)	4,227	866	5,093
Adjective phrase (ADJP)	2,060	438	2,498
Subordinated clause (SBAR)	2,207	535	2,742
Particles (PRT)	556	106	662
Conjunction phrase (CONJP)	56	9	65
Interjection (INTJ)	31	2	33
List marker (LST)	10	5	15
Unlike coordinated phrase (UCP)	2	0	2
Outside of any chunk (O)	27,902	6,180	34,082

Table B.4: Statistics of the UD English EWT data set (version 2.5; Section 3.4.1). The training, development (Devel.), and test sets were included, as well as the number of sentences and tokens.

Element	Training	Devel.	Test	Total
Sentences	12,543	2,002	2,077	16,622
Tokens	204,585	25,148	25,096	254,829
Adjective (ADJ)	12,482	1,788	1,691	15,961
Adposition (ADP)	17,625	2,021	2,020	21,666
Adverb (ADV)	10,559	1,265	1,226	13,050
Auxiliary (AUX)	12,396	1,504	1,512	15,412
Coordinating conjunction (CCONJ)	6,703	780	738	8,221
Determiner (DET)	16,284	1,895	1,896	20,075
Interjection (INTJ)	688	115	120	923
Noun (NOUN)	34,740	4,192	4,127	43,059
Numeral (NUM)	3,996	378	536	4,910
Particle (PART)	5,567	630	630	6,827
Pronoun (PRON)	18,579	2,218	2,158	22,955
Proper noun (PROPN)	12,944	1,879	2,075	16,898
Punctuation (PUNCT)	23,676	3,083	3,106	29,865
Subordinating conjunction (SCONJ)	3,850	403	386	4,639
Symbol (SYM)	643	75	100	818
Verb (VERB)	23,006	2,759	2,644	28,409
Other (X)	847	155	139	1,141

Table B.5: Statistics of the UD English EWT data set (version 2.6; Section 4.5.5). The training, development (Devel.), and test sets were included, as well as the number of sentences and tokens.

Element	Training	Devel.	Test	Total
Sentences	12,543	2,002	2,077	16,622
Tokens	204,585	25,148	25,096	254,829
Adjective (ADJ)	12,458	1,784	1,689	15,931
Adposition (ADP)	17,625	2,021	2,020	21,666
Adverb (ADV)	10,553	1,264	1,226	13,043
Auxiliary (AUX)	12,396	1,512	1,504	15,412
Coordinating conjunction (CCONJ)	6,703	780	738	8,221
Determiner (DET)	16,284	1,895	1,896	20,075
Interjection (INTJ)	688	115	120	923
Noun (NOUN)	34,765	4,196	4,129	43,090
Numeral (NUM)	3,996	378	536	4,910
Particle (PART)	5,567	630	630	6,827
Pronoun (PRON)	18,584	2,219	2,158	22,961
Proper noun (PROPN)	12,945	1,879	2,075	16,899
Punctuation (PUNCT)	23,676	3,083	3,106	29,865
Subordinating conjunction (SCONJ)	3,850	403	386	4,639
Symbol (SYM)	643	75	100	818
Verb (VERB)	23,005	2,759	2,644	28,408
other (X)	847	155	139	1,141

Table B.6: Statistics of the UD German GSD data set (version 2.5; Section 3.4.1). The training, development (Devel.), and test sets were included, as well as the number of sentences and tokens.

Element	Training	Devel.	Test	Total
Sentences	13,814	799	977	15,590
Tokens	263,804	12,486	16,498	292,788
Adjective (ADJ)	18,741	807	1,020	20,568
Adposition (ADP)	29,266	1,119	1,604	31,989
Adverb (ADV)	12,456	1,123	1,283	14,862
Auxiliary (AUX)	9,272	635	687	10,594
Coordinating conjunction (CCONJ)	7,977	418	464	8,859
Determiner (DET)	33,594	1,314	1,888	36,796
Interjection (INTJ)	0	1	4	5
Noun (NOUN)	46,895	2,197	3,110	52,202
Numeral (NUM)	7,007	180	266	7,453
Particle (PART)	1,794	148	211	2,153
Pronoun (PRON)	12,980	1,018	1,043	15,041
Proper noun (PROPN)	29,218	614	1,025	30,857
Punctuation (PUNCT)	34,470	1,677	2,366	38,513
Subordinating conjunction (SCONJ)	1,466	115	167	1,748
Symbol (SYM)	86	2	4	92
Verb (VERB)	18,286	1,096	1,330	20,712
Other (X)	296	22	26	344

C Incorporated Publications

C.1 "Efficient, Lexicon-Free OCR using Deep Learning"

This publication constitutes the core part of Chapter 2.

©2019 IEEE. Reprinted, with permission, from M. Namysl and I. Konya (2019). "Efficient, lexicon-free OCR using deep learning." In: *International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, pp. 295–301. DOI: 10.1109/ICDAR.2019.00055.

C.2 "NAT: Noise-Aware Training for Robust Neural Sequence Labeling"

This publication constitutes the core part of Chapter 3.

©2020 Association for Computational Linguistics. Reprinted under the terms of Creative Commons Attribution 4.0 International License (https://creativecommons.org/licenses/by/4.0) from M. Namysl, S. Behnke, J. Köhler (2020). "NAT: Noise-aware training for robust neural sequence labeling." In: *Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 1501–1517. DOI: 10.18653/v1/2020.acl-main.138.

C.3 “Empirical Error Modeling Improves Robustness of Noisy Neural Sequence Labeling”

This publication constitutes the core part of Chapter 4.
©2021 Association for Computational Linguistics. Reprinted under the terms of Creative Commons Attribution 4.0 International License (https://creativecommons.org/licenses/by/4.0) from M. Namysl, S. Behnke, J. Köhler (2021). “Empirical error modeling improves robustness of noisy neural sequence labeling.” In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP*. Association for Computational Linguistics, pp. 314–329. DOI: 10.18653/v1/2021.findings-acl.27.

C.4 “Flexible Table Recognition and Semantic Interpretation System”

This publication constitutes the core part of Chapter 5.
©2022 SciTePress. Reprinted, with permission, from M. Namysl, A. Esser, S. Behnke, J. Köhler (2022). “Flexible table recognition and semantic interpretation system.” In: *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. Vol. 4: VISAPP. INSTICC. SciTePress, pp. 27–37. ISBN: 978-989-758-555-5. DOI: 10.5220/0010767600003124.

C.5 “Benchmarking table recognition performance on biomedical literature on neurological disorders”

This publication constitutes the core part of Chapter 6.
©2021 Oxford University Press. Reproduced by permission of Oxford University Press (https://academic.oup.com). Reprinted from T. Adams, M. Namysl, A. T. Kodamullil, S. Behnke, and M. Jacobs (2021). “Benchmarking table recognition performance on biomedical literature on neurological disorders.” In: *Bioinformatics* 38.6, pp. 1624–1630. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btab843.

Acronyms

AEG artificial error generation 75, 78, 79

ASR automatic speech recognition 3, 40, 42, 44, 65, 69, 72, 80, 82

BLEU bilingual evaluation understudy 148, 161

BLSTM bidirectional long short-term memory 17, 18, 20–22, 32, 35, 37, 51, 52, 167, 171

BRNN bidirectional recurrent neural network 13, 17, 89, 171

CER character error rate 30–32

CNN convolutional neural network 10, 12, 14, 15, 17–22, 32, 35, 37, 110, 167

CPU central processing unit 34, 36, 38, 86

CRF conditional random field 42, 43, 51, 52

CRNN convolutional recurrent neural network 18, 20

CTC connectionist temporal classification 18–22, 35, 37, 167, 168

CTR complete table recognition 6, 8, 109, 129, 132, 135, 146, 149, 150, 154, 157–159, 161, 164, 165

DPI dots per inch 132

FCN fully convolutional network 18, 20–22, 30, 32, 35, 37, 38, 168

GEC grammatical error correction 74, 75, 79

GPU graphics processing unit 34

GRU gated recurrent unit 13, 32, 36, 38

HDAC histone deacetylase 124, 125, 139

HMM hidden Markov model 17

HOG histogram of oriented gradients 16

HTML HyperText Markup Language 110, 146, 148–150, 164

HTR handwritten text recognition 17–20

ICDAR International Conference on Document Analysis and Recognition 104–106, 115, 118, 122, 128, 130–138, 142, 147, 149, 171

IE information extraction v, 1–3, 6, 8, 40, 72, 77, 104, 106, 113, 126, 141–143, 163, 164

IoU intersection over union 136–138, 147, 160

JSON JavaScript Object Notation 124, 125, 139–141, 148

LSTM long short-term memory 12, 13, 18, 19, 30, 32, 36, 38, 167

MDRNN multidimensional recurrent neural network 14, 17, 18

ML machine learning 1, 68, 112

NAT noise-aware training v, 4, 5, 8, 40–42, 45, 50, 52, 54, 57, 59, 62–64, 67–69, 72, 73, 75, 76, 78, 81, 85, 89, 91, 92, 94, 95, 98, 99, 101, 163–165

NER named entity recognition v, 2, 3, 40, 44, 47, 50, 51, 54–57, 65, 85, 86, 94

NLM noisy language modeling v, 5, 8, 73, 82, 85, 89, 94–97, 101, 163–165

NLP natural language processing 2, 3, 8, 19, 41, 65, 72, 75, 77, 101, 111, 164

NMT neural machine translation 19, 42, 65–67, 74, 75

OCR optical character recognition v, 2–5, 10, 11, 16–20, 27, 28, 30–33, 36, 40, 42, 44, 45, 47, 52–56, 58–61, 63, 65, 72–74, 77–80, 82, 84, 85, 87–91, 95–99, 101, 114, 148, 159, 163, 170

PDF Portable Document Format 1, 3, 5, 103, 105, 106, 110, 112–114, 126, 127, 132–134, 139, 142, 146–149, 152, 154, 155, 159

PMC PubMed Central 146, 148, 149, 152

POST part-of-speech tagging v, 40, 51, 52, 57–59, 85, 86

RegEx regular expressions 5, 105, 111–113, 123

RNN recurrent neural network 10, 12–14, 17–19, 32

STR scene text recognition 11, 18–20, 28, 164

TSR table structure recognition 5, 103, 106–108, 110–112, 126–128, 130–134, 142, 148, 149, 171

XML Extensible Markup Language 27, 134, 137, 146, 148, 149, 152, 155, 159

Bibliography

Abadi, M., P. Barham, J. Chen, Z. Chen, A. Davis, et al. (2016). “TensorFlow: A system for large-scale machine learning.” In: *USENIX Conference on Operating Systems Design and Implementation (OSDI)*. USENIX Association, pp. 265–283. ISBN: 9781931971331. URL: https://dl.acm.org/doi/10.5555/3026877.3026899 (cit. on p. 29).

Adams, T. and M. Namysl (2021). *Table recognition benchmark on biomedical literature on neurological disorders*. Version 1.0. Zenodo. DOI: 10.5281/zenodo.5549977 (cit. on pp. 6, 8, 146, 147, 148, 149, 152, 160).

Adams, T., M. Namysl, A. T. Kodamullil, S. Behnke, and M. Jacobs (2021). “Benchmarking table recognition performance on biomedical literature on neurological disorders.” In: *Bioinformatics* 38.6, pp. 1624–1630. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btab843 (cit. on pp. 7, 8, 137, 145, 180).

Afli, H., Z. Qiu, A. Way, and P. Sheridan (2016). “Using SMT for OCR error correction of historical texts.” In: *International Conference on Language Resources and Evaluation (LREC)*. European Language Resources Association (ELRA), pp. 962–966. URL: https://aclanthology.org/L16-1153 (cit. on p. 74).

Akbik, A., T. Bergmann, D. Blythe, K. Rasul, S. Schweter, and R. Vollgraf (2019). “FLAIR: An easy-to-use framework for state-of-the-art NLP.” In: *Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. Association for Computational Linguistics, pp. 54–59. DOI: 10.18653/v1/N19-4010 (cit. on pp. 54, 84).

Akbik, A., T. Bergmann, and R. Vollgraf (2019). “Pooled contextualized embeddings for named entity recognition.” In: *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, pp. 724–728. DOI: 10.18653/v1/N19-1078 (cit. on p. 72).

Akbik, A., D. Blythe, and R. Vollgraf (2018). “Contextual string embeddings for sequence labeling.” In: *International Conference on Computational Linguistics*. Association for Computational Linguistics, pp. 1638–1649. URL: https://aclanthology.org/C18-1139 (cit. on pp. 42, 51, 52, 73, 77, 78, 82, 85, 94).

Al Azawi, M., M. Liwicki, and T. M. Breuel (2015). “Combination of multiple aligned recognition outputs using WFST and LSTM.” In: *International Conference on Document Analysis and Recognition (ICDAR)*, pp. 31–35. DOI: 10.1109/ICDAR.2015.7333720 (cit. on p. 74).

Alex, B. and J. Burns (2014). “Estimating and rating the quality of optically character recognised text.” In: *International Conference on Digital Access to Textual Cultural Heritage (DATeCH)*. ACM, pp. 97–102. ISBN: 978-1-4503-2588-2. DOI: 10.1145/2595188.2595214 (cit. on pp. 40, 44, 65, 72, 94).

Allison, B., D. Guthrie, and L. Guthrie (2006). “Another look at the data sparsity problem.” In: *International Conference on Text, Speech and Dialogue (TSD)*. Springer-Verlag, pp. 327–334. ISBN: 3540390901. DOI: 10.1007/11846406_41 (cit. on p. 3).

Alzantot, M., Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang (2018). “Generating natural language adversarial examples.” In: *Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 2890–2896. DOI: 10.18653/v1/D18-1316 (cit. on p. 68).

Asad, F., A. Ul-Hasan, F. Shafait, and A. Dengel (2016). “High performance OCR for camera-captured blurred documents with LSTM networks.” In: *IAPR Workshop on Document Analysis Systems (DAS)*, pp. 7–12. DOI: 10.1109/DAS.2016.69 (cit. on p. 17).

Auer, S., C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives (2007). “DBpedia: A nucleus for a web of open data.” In: *The Semantic Web*. Springer Berlin Heidelberg, pp. 722–735. ISBN: 978-3-540-76298-0. DOI: 10.1007/978-3-540-76298-0_52 (cit. on p. 112).

Bahdanau, D., K. Cho, and Y. Bengio (2015). “Neural machine translation by jointly learning to align and translate.” In: *International Conference on Learning Representations (ICLR), Conference Track Proceedings*. URL: http://arxiv.org/abs/1409.0473 (cit. on pp. 19, 78).

Bailey, D. G. (2011). “Image border management for FPGA based filters.” In: *IEEE International Symposium on Electronic Design, Test and Application*, pp. 144–149. DOI: 10.1109/DELTA.2011.34 (cit. on p. 24).

Baldwin, T., P. Cook, M. Lui, A. MacKinlay, and L. Wang (2013). “How noisy social media text, how diffrnt social media sources?” In: *International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing, pp. 356–364. URL: https://aclanthology.org/I13-1041 (cit. on p. 72).

Bekoulis, G., J. Deleu, T. Demeester, and C. Develder (2018). “Adversarial training for multi-context joint entity and relation extraction.” In: *Conference on Em-*

pirical Methods in Natural Language Processing. Association for Computational Linguistics, pp. 2830–2836. DOI: 10.18653/v1/D18-1307 (cit. on pp. 47, 68).

Belinkov, Y. and Y. Bisk (2018). "Synthetic and natural noise both break neural machine translation." In: *International Conference on Learning Representations (ICLR), Conference Track Proceedings.* URL: https://openreview.net/forum?id=BJ8vJebC- (cit. on pp. 41, 53, 55, 56, 58, 59, 66, 72, 87, 88).

Benikova, D., C. Biemann, and M. Reznicek (2014). "NoSta-D named entity annotation for German: Guidelines and dataset." In: *International Conference on Language Resources and Evaluation (LREC).* European Language Resources Association (ELRA), pp. 2524–2531. URL: http://www.lrec-conf.org/proceedings/lrec2014/pdf/276_Paper.pdf (cit. on pp. 47, 50).

Biesner, D., R. Ramamurthy, R. Stenzel, M. Lübbering, L. Hillebrand, A. Ladi, M. Pielka, R. Loitz, C. Bauckhage, and R. Sifa (2022). "Anonymization of German financial documents using neural network-based language models with contextual word representations." In: *International Journal of Data Science and Analytics* 13.2, pp. 151–161. ISSN: 2364-4168. DOI: 10.1007/s41060-021-00285-x (cit. on p. 69).

Bissacco, A., M. Cummins, Y. Netzer, and H. Neven (2013). "PhotoOCR: Reading text in uncontrolled conditions." In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 785–792. DOI: 10.1109/ICCV.2013.102 (cit. on p. 16).

Bojanowski, P., E. Grave, A. Joulin, and T. Mikolov (2017). "Enriching word vectors with subword information." In: *Transactions of the Association for Computational Linguistics* 5, pp. 135–146. DOI: 10.1162/tacl_a_00051 (cit. on p. 51).

Borisyuk, F., A. Gordo, and V. Sivakumar (2018). "Rosetta: Large scale system for text detection and recognition in images." In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, pp. 71–79. ISBN: 978-1-4503-5552-0. DOI: 10.1145/3219819.3219861 (cit. on pp. 18, 20).

Breuel, T. M. (2017). "High performance text recognition using a hybrid convolutional-LSTM implementation." In: *IAPR International Conference on Document Analysis and Recognition (ICDAR).* Vol. 01, pp. 11–16. DOI: 10.1109/ICDAR.2017.12 (cit. on pp. 18, 25).

Breuel, T. M., A. Ul-Hasan, M. A. Al-Azawi, and F. Shafait (2013). "High-performance OCR for printed English and fraktur using LSTM networks." In: *International Conference on Document Analysis and Recognition (ICDAR)*, pp. 683–687. DOI: 10.1109/ICDAR.2013.140 (cit. on p. 17).

Breuel, T. M. (2008). "The OCRopus open source OCR system." In: *Document Recognition and Retrieval XV*. Vol. 6815. International Society for Optics and Photonics. SPIE, pp. 120–134. DOI: 10.1117/12.783598 (cit. on p. 16).

Bridle, J. S. (1990). "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition." In: *Neurocomputing*. Springer Berlin Heidelberg, pp. 227–236. ISBN: 978-3-642-76153-9. DOI: 10.1007/978-3-642-76153-9_28 (cit. on pp. 15, 20).

Brill, E. and R. C. Moore (2000). "An improved error model for noisy channel spelling correction." In: *Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 286–293. DOI: 10.3115/1075218.1075255 (cit. on pp. 2, 44, 74).

Brown, T., B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, et al. (2020). "Language models are few-shot learners." In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., pp. 1877–1901. URL: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf (cit. on p. 77).

Bryant, C., M. Felice, Ø. E. Andersen, and T. Briscoe (2019). "The BEA-2019 shared task on grammatical error correction." In: *Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, pp. 52–75. DOI: 10.18653/v1/W19-4406 (cit. on p. 74).

Bunke, H., S. Bengio, and A. Vinciarelli (2004). "Offline recognition of unconstrained handwritten texts using HMMs and statistical language models." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.6, pp. 709–720. DOI: 10.1109/TPAMI.2004.14 (cit. on p. 17).

Bušta, M., L. Neumann, and J. Matas (2017). "Deep TextSpotter: An end-to-end trainable scene text localization and recognition framework." In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2223–2231. DOI: 10.1109/ICCV.2017.242 (cit. on pp. 11, 18).

Chauhan, R., K. K. Ghanshala, and R. Joshi (2018). "Convolutional neural network (CNN) for image detection and recognition." In: *International Conference on Secure Cyber Computing and Communication (ICSCCC)*, pp. 278–282. DOI: 10.1109/ICSCCC.2018.8703316 (cit. on p. 14).

Chen, P.-J., I.-H. Hsu, Y. Y. Huang, and H.-Y. Lee (2017). "Mitigating the impact of speech recognition errors on chatbot using sequence-to-sequence model." In: *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 497–503. DOI: 10.1109/ASRU.2017.8268977 (cit. on pp. 44, 65).

Cheng, Y., L. Jiang, and W. Macherey (2019). "Robust neural machine translation with doubly adversarial inputs." In: *Annual Meeting of the Association for*

Computational Linguistics. Association for Computational Linguistics, pp. 4324–4333. DOI: 10.18653/v1/P19-1425 (cit. on p. 59).

Cheng, Y., Z. Tu, F. Meng, J. Zhai, and Y. Liu (2018). "Towards robust neural machine translation." In: *Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pp. 1756–1766. DOI: 10.18653/v1/P18-1163 (cit. on pp. 42, 67).

Cheung, M., J. Shi, O. Wright, L. Y. Jiang, X. Liu, and J. M. F. Moura (2020). "Graph signal processing and deep learning: convolution, pooling, and topology." In: *IEEE Signal Processing Magazine* 37.6, pp. 139–149. DOI: 10.1109/MSP.2020.3014594 (cit. on p. 15).

Chi, Z., H. Huang, H.-D. Xu, H. Yu, W. Yin, and X.-L. Mao (2019). *Complicated table structure recognition*. DOI: 10.48550/ARXIV.1908.04729 (cit. on pp. 3, 146, 148, 153).

Chiron, G., A. Doucet, M. Coustaty, and J. Moreux (2017). "ICDAR2017 competition on post-OCR text correction." In: *IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 01, pp. 1423–1428. DOI: 10.1109/ICDAR.2017.232 (cit. on pp. 74, 78).

Chiu, J. P. and E. Nichols (2016). "Named entity recognition with bidirectional LSTM-CNNs." In: *Transactions of the Association for Computational Linguistics* 4, pp. 357–370. DOI: 10.1162/tacl_a_00104 (cit. on p. 42).

Cho, K., B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio (2014). "Learning phrase representations using RNN encoder–decoder for statistical machine translation." In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pp. 1724–1734. DOI: 10.3115/v1/D14-1179 (cit. on p. 13).

Choe, Y. J., J. Ham, K. Park, and Y. Yoon (2019). "A neural grammatical error correction system built on better pre-training and sequential transfer learning." In: *Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, pp. 213–227. DOI: 10.18653/v1/W19-4423 (cit. on p. 75).

Chowdhury, A. and L. Vig (2018). *An efficient end-to-end neural model for handwritten text recognition*. DOI: 10.48550/ARXIV.1807.07965 (cit. on p. 19).

Coşkun, M., A. Uçar, Ö. Yildirim, and Y. Demir (2017). "Face recognition based on convolutional neural network." In: *International Conference on Modern Electrical and Energy Systems (MEES)*, pp. 376–379. DOI: 10.1109/MEES.2017.8248937 (cit. on p. 15).

Cubuk, E. D., B. Zoph, D. Mané, V. Vasudevan, and Q. V. Le (2019). "AutoAugment: Learning augmentation strategies from data." In: *IEEE/CVF Conference* *on Computer Vision and Pattern Recognition (CVPR)*, pp. 113–123. DOI: 10.1109/CVPR.2019.00020 (cit. on pp. 65, 68).

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei (2009). “ImageNet: A large-scale hierarchical image database.” In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848 (cit. on p. 133).

Deng, Y., A. Kanervisto, J. Ling, and A. M. Rush (2017). “Image-to-markup generation with coarse-to-fine attention.” In: *International Conference on Machine Learning (ICML)*. Vol. 70. PMLR, pp. 980–989. URL: https://proceedings.mlr.press/v70/deng17a.html (cit. on p. 2).

Derczynski, L., A. Ritter, S. Clark, and K. Bontcheva (2013). “Twitter part-of-speech tagging for all: Overcoming sparse and noisy data.” In: *International Conference on Recent Advances in Natural Language Processing (RANLP)*. INCOMA Ltd., pp. 198–206. URL: https://aclanthology.org/R13-1026 (cit. on pp. 40, 44, 72).

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2019). “BERT: Pre-training of deep bidirectional transformers for language understanding.” In: *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, pp. 4171–4186. DOI: 10.18653/v1/N19-1423 (cit. on pp. 19, 42, 52, 72, 77).

Dhole, K. D., V. Gangal, S. Gehrmann, A. Gupta, Z. Li, et al. (2021). *NL-Augmenter: A framework for task-sensitive natural language augmentation.* DOI: 10.48550/ARXIV.2112.02721 (cit. on pp. 7, 8).

Diaz, D. H., S. Qin, R. Ingle, Y. Fujii, and A. Bissacco (2021). *Rethinking text line recognition models.* DOI: 10.48550/ARXIV.2104.07787 (cit. on p. 19).

Diem, M., F. Kleber, and R. Sablatnig (2013). “Text line detection for heterogeneous documents.” In: *International Conference on Document Analysis and Recognition (ICDAR)*, pp. 743–747. DOI: 10.1109/ICDAR.2013.152 (cit. on p. 2).

Dosovitskiy, A., L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, et al. (2021). “An image is worth 16x16 words: Transformers for image recognition at scale.” In: *International Conference on Learning Representations (ICLR)*. URL: https://openreview.net/forum?id=YicbFdNTTy (cit. on p. 19).

Duan, H. and B.-J. Hsu (2011). “Online spelling correction for query completion.” In: *International Conference on World Wide Web (WWW)*. Association for Computing Machinery, pp. 117–126. ISBN: 9781450306324. DOI: 10.1145/1963405.1963425 (cit. on p. 74).

Ebrahimi, J., A. Rao, D. Lowd, and D. Dou (2018). "HotFlip: White-box adversarial examples for text classification." In: *Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, pp. 31–36. DOI: 10.18653/v1/P18-2006 (cit. on p. 68).

Edmonds, J. (1965). "Maximum matching and a polyhedron with 0, 1 vertices." In: *Journal of Research of the National Bureau of Standards - B. Mathematics and Mathematical Physics* 69 B, pp. 125–130. URL: https://nvlpubs.nist.gov/nistpubs/jres/69B/jresv69Bn1-2p125_A1b.pdf (cit. on pp. 5, 105, 107, 124, 141, 152).

Edunov, S., M. Ott, M. Auli, and D. Grangier (2018). "Understanding back-translation at scale." In: *Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 489–500. DOI: 10.18653/v1/D18-1045 (cit. on p. 75).

Evans, P., A. Sherin, and I. Lee (2013). *The graphic design reference and specification book: Everything graphic designers need to know every day*. Rockport Publishers. ISBN: 9781610587884. URL: https://books.google.de/books?id=4rXWAgAAQBAJ (cit. on p. 25).

Fischer, P., A. Smajic, G. Abrami, and A. Mehler (2021). "Multi-Type-TD-TSR - Extracting tables from document images using a multi-stage pipeline for table detection and table structure recognition: From OCR to structured table representations." In: *KI 2021: Advances in Artificial Intelligence*. Vol. 12873. Springer International Publishing, pp. 95–108. ISBN: 978-3-030-87626-5. DOI: 10.1007/978-3-030-87626-5_8 (cit. on pp. 111, 137).

Fivez, P., S. Šuster, and W. Daelemans (2017). "Unsupervised context-sensitive spelling correction of clinical free-text with word and character n-gram embeddings." In: *BioNLP*. Association for Computational Linguistics, pp. 143–148. DOI: 10.18653/v1/W17-2317 (cit. on p. 74).

Flor, M., M. Fried, and A. Rozovskaya (2019). "A benchmark corpus of English misspellings and a minimally-supervised model for spelling correction." In: *Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, pp. 76–86. DOI: 10.18653/v1/W19-4407 (cit. on pp. 74, 78).

Gales, M. and S. Young (2007). "The application of hidden Markov models in speech recognition." In: *Foundations and Trends in Signal Processing* 1.3, pp. 195–304. ISSN: 1932-8346. DOI: 10.1561/2000000004 (cit. on p. 17).

Gao, J., J. Lanchantin, M. L. Soffa, and Y. Qi (2018). "Black-box generation of adversarial text sequences to evade deep learning classifiers." In: *IEEE Security and Privacy Workshops (SPW)*, pp. 50–56. DOI: 10.1109/SPW.2018.00016 (cit. on p. 68).

Gao, L., Y. Huang, H. Déjean, J.-L. Meunier, Q. Yan, Y. Fang, F. Kleber, and E. Lang (2019). "ICDAR 2019 competition on table detection and recognition (cTDaR)." In: *International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1510–1515. DOI: 10.1109/ICDAR.2019.00243 (cit. on pp. 3, 105, 115, 130, 131, 132, 138, 147).

Gatos, B., D. Danatsas, I. Pratikakis, and S. J. Perantonis (2005). "Automatic table detection in document images." In: *Pattern Recognition and Data Mining*. Springer Berlin Heidelberg, pp. 609–618. ISBN: 978-3-540-28758-2 (cit. on p. 114).

Gers, F. A. and J. Schmidhuber (2000). "Recurrent nets that time and count." In: *IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN). Neural Computing: New Challenges and Perspectives for the New Millennium*. Vol. 3, pp. 189–194. DOI: 10.1109/IJCNN.2000.861302 (cit. on p. 13).

Ghosh, S. K., E. Valveny, and A. D. Bagdanov (2017). "Visual attention models for scene text recognition." In: *IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 01, pp. 943–948. DOI: 10.1109/ICDAR.2017.158 (cit. on p. 19).

Göbel, M., T. Hassan, E. Oro, and G. Orsi (2013). "ICDAR 2013 Table Competition." In: *International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1449–1453. DOI: 10.1109/ICDAR.2013.292 (cit. on pp. 3, 105, 118, 122, 128, 132, 134, 135, 147, 154).

Göbel, M., T. Hassan, E. Oro, and G. Orsi (2012). "A methodology for evaluating algorithms for table understanding in PDF documents." In: *ACM Symposium on Document Engineering (DocEng)*. Association for Computing Machinery, pp. 45–48. ISBN: 9781450311168. DOI: 10.1145/2361354.2361365 (cit. on pp. 147, 150, 154).

Goldhahn, D., T. Eckart, and U. Quasthoff (2012). "Building large monolingual dictionaries at the Leipzig Corpora Collection: From 100 to 200 languages." In: *Language Resources and Evaluation Conference (LREC)*. Vol. 29. European Language Resources Association (ELRA), pp. 759–765. URL: http://www.lrec-conf.org/proceedings/lrec2012/pdf/327_Paper.pdf (cit. on p. 27).

Goodfellow, I., J. Shlens, and C. Szegedy (2015). "Explaining and harnessing adversarial examples." In: *International Conference on Learning Representations (ICLR)*. URL: http://arxiv.org/abs/1412.6572 (cit. on pp. 1, 68).

Graves, A., M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber (2009). "A novel connectionist system for unconstrained handwriting recognition." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.5, pp. 855–868. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2008.137 (cit. on p. 17).

Graves, A., S. Fernández, F. Gomez, and J. Schmidhuber (2006). "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks." In: *International Conference on Machine Learning (ICML)*. ACM, pp. 369–376. ISBN: 1-59593-383-2. DOI: 10.1145/1143844.1143891 (cit. on p. 18).

Graves, A., S. Fernández, and J. Schmidhuber (2007). "Multi-dimensional recurrent neural networks." In: *Artificial Neural Networks (ICANN)*. Springer Berlin Heidelberg, pp. 549–558. ISBN: 978-3-540-74690-4. DOI: 10.1007/978-3-540-74690-4_56 (cit. on p. 14).

Graves, A., A.-r. Mohamed, and G. Hinton (2013). "Speech recognition with deep recurrent neural networks." In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6645–6649. DOI: 10.1109/ICASSP.2013.6638947 (cit. on pp. 12, 13, 14).

Graves, A. and J. Schmidhuber (2008). "Offline handwriting recognition with multidimensional recurrent neural networks." In: *International Conference on Neural Information Processing Systems*. Curran Associates Inc., pp. 545–552. ISBN: 978-1-6056-0-949-2. URL: https://proceedings.neurips.cc/paper/2008/file/66368270ffd51418ec58bd793f2d9b1b-Paper.pdf (cit. on p. 17).

Grundkiewicz, R. and M. Junczys-Dowmunt (2019). "Minimally-augmented grammatical error correction." In: *Workshop on Noisy User-generated Text (W-NUT)*. Association for Computational Linguistics, pp. 357–363. DOI: 10.18653/v1/D19-5546 (cit. on p. 101).

Grundkiewicz, R., M. Junczys-Dowmunt, and K. Heafield (2019). "Neural grammatical error correction systems with unsupervised pre-training on synthetic data." In: *Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, pp. 252–263. DOI: 10.18653/v1/W19-4427 (cit. on p. 75).

Hämäläinen, M. and S. Hengchen (2019). "From the paft to the fiiture: A fully automatic NMT and word embeddings method for OCR post-correction." In: *International Conference on Recent Advances in Natural Language Processing (RANLP)*. INCOMA Ltd., pp. 431–436. DOI: 10.26615/978-954-452-056-4_051 (cit. on pp. 74, 89).

Haralick, R. M. and L. G. Shapiro (1985). "Image segmentation techniques." In: *Applications of Artificial Intelligence II*. Vol. 0548. International Society for Optics and Photonics. SPIE, pp. 2–9. DOI: 10.1117/12.948400 (cit. on p. 107).

Hassan, T. and R. Baumgartner (2007). "Table recognition and understanding from PDF files." In: *International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 2, pp. 1143–1147. DOI: 10.1109/ICDAR.2007.4377094 (cit. on p. 110).

He, K., X. Zhang, S. Ren, and J. Sun (2016). “Deep residual learning for image recognition.” In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778. DOI: 10.1109/CVPR.2016.90 (cit. on p. 10).

Heigold, G., S. Varanasi, G. Neumann, and J. van Genabith (2018). “How robust are character-based word embeddings in tagging and MT against wrod scramlbing or randdm nouse?” In: *Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*. Association for Machine Translation in the Americas, pp. 68–80. URL: https://aclanthology.org/W18-1807 (cit. on pp. 59, 66, 72, 87).

Heinzerling, B. and M. Strube (2019). “Sequence tagging with contextual and non-contextual subword representations: A multilingual evaluation.” In: *Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 273–291. DOI: 10.18653/v1/P19-1027 (cit. on p. 72).

Hochreiter, S. and J. Schmidhuber (1997). “Long short-term memory.” In: *Neural Computation* 9.8, pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735 (cit. on p. 12).

Hoshen, J. and R. Kopelman (1976). “Percolation and cluster distribution. I. Cluster multiple labeling technique and critical concentration algorithm.” In: *Physical Review B* 14 (8), pp. 3438–3445. DOI: 10.1103/PhysRevB.14.3438 (cit. on p. 116).

Hu, J., L. Shen, and G. Sun (2018). “Squeeze-and-excitation networks.” In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7132–7141. DOI: 10.1109/CVPR.2018.00745 (cit. on pp. 10, 38).

Huang, Y., T. Lv, L. Cui, Y. Lu, and F. Wei (2022). *LayoutLMv3: Pre-training for document AI with unified text and image masking.* DOI: 10.48550/ARXIV.2204.08387 (cit. on p. 143).

Huang, Z., M. Dong, Q. Mao, and Y. Zhan (2014). “Speech emotion recognition using CNN.” In: *ACM International Conference on Multimedia*. Association for Computing Machinery, pp. 801–804. ISBN: 9781450330633. DOI: 10.1145/2647868.2654984 (cit. on p. 14).

Huang, Z., W. Xu, and K. Yu (2015). *Bidirectional LSTM-CRF models for sequence tagging.* DOI: 10.48550/ARXIV.1508.01991 (cit. on p. 51).

Hubel, D. H. and T. N. Wiesel (1962). “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex.” In: *The Journal of Physiology* 160.1, pp. 106–154. DOI: 10.1113/jphysiol.1962.sp006837 (cit. on p. 14).

Hulsebos, M., K. Hu, M. Bakker, E. Zgraggen, A. Satyanarayan, T. Kraska, Ç. Demiralp, and C. Hidalgo (2019). “Sherlock: A deep learning approach to semantic data type detection.” In: *ACM SIGKDD International Conference* *on Knowledge Discovery and Data Mining (KDD)*. Association for Computing Machinery, pp. 1500–1508. ISBN: 9781450362016. DOI: 10.1145/3292500.3330993 (cit. on p. 112).

IEEE (1990). "IEEE standard glossary of software engineering terminology." In: *IEEE Std 610.12-1990*, pp. 1–84. DOI: 10.1109/IEEESTD.1990.101064 (cit. on p. 1).

Ioffe, S. and C. Szegedy (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift." In: *International Conference on Machine Learning (ICML)*. Vol. 37. PMLR, pp. 448–456. URL: https://proceedings.mlr.press/v37/ioffe15.html (cit. on pp. 10, 30, 167, 168).

Jacobs, C., P. Y. Simard, P. Viola, and J. Rinker (2005). "Text recognition of low-resolution document images." In: *International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 2, pp. 695–699. DOI: 10.1109/ICDAR.2005.233 (cit. on p. 16).

Jaderberg, M., K. Simonyan, A. Vedaldi, and A. Zisserman (2014). *Synthetic data and artificial neural networks for natural scene text recognition*. DOI: 10.48550/ARXIV.1406.2227 (cit. on pp. 18, 22, 24, 38).

Jaderberg, M., K. Simonyan, A. Zisserman, and K. Kavukcuoglu (2015). "Spatial transformer networks." In: *Advances in Neural Information Processing Systems*. Vol. 28. Curran Associates, Inc., pp. 2017–2025. URL: https://proceedings.neurips.cc/paper/2015/file/33ceb07bf4eeb3da587e268d663aba1a-Paper.pdf (cit. on p. 38).

Ji, H. and R. Grishman (2011). "Knowledge base population: Successful approaches and challenges." In: *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pp. 1148–1158. URL: https://aclanthology.org/P11-1115 (cit. on p. 2).

Jia, R., A. Raghunathan, K. Göksel, and P. Liang (2019). "Certified robustness to adversarial word substitutions." In: *Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, pp. 4129–4142. DOI: 10.18653/v1/D19-1423 (cit. on p. 67).

Jimeno Yepes, A. and K. Verspoor (2014). "Literature mining of genetic variants for curation: Quantifying the importance of supplementary material." In: *Database* 2014. ISSN: 1758-0463. DOI: 10.1093/database/bau003 (cit. on p. 146).

Jivani, A. G. et al. (2011). "A comparative study of stemming algorithms." In: *International Journal of Computer Technology and Applications* 2.6, pp. 1930–1938. ISSN: 2229-6093 (cit. on p. 2).

Jones, E., R. Jia, A. Raghunathan, and P. Liang (2020). "Robust encodings: A framework for combating adversarial typos." In: *Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 2752–2765. DOI: 10.18653/v1/2020.acl-main.245 (cit. on p. 67).

Kagaya, H., K. Aizawa, and M. Ogawa (2014). "Food detection and recognition using convolutional neural network." In: *ACM International Conference on Multimedia*. Association for Computing Machinery, pp. 1085–1088. ISBN: 9781450330633. DOI: 10.1145/2647868.2654970 (cit. on p. 15).

Karpukhin, V., O. Levy, J. Eisenstein, and M. Ghazvininejad (2019). "Training on synthetic noise improves robustness to natural noise in machine translation." In: *Workshop on Noisy User-generated Text (W-NUT)*. Association for Computational Linguistics, pp. 42–47. DOI: 10.18653/v1/D19-5506 (cit. on pp. 41, 66).

Kasewa, S., P. Stenetorp, and S. Riedel (2018). "Wronging a right: Generating better errors to improve grammatical error detection." In: *Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 4977–4983. DOI: 10.18653/v1/D18-1541 (cit. on p. 75).

Kemighan, M. D., K. W. Church, and W. A. Gale (1990). "A spelling correction program based on a noisy channel model." In: *International Conference on Computational Linguistics (COLING)*. URL: https://aclanthology.org/C90-2036 (cit. on p. 74).

Kim, S. M. and S. Cassidy (2015). "Finding names in Trove: Named entity recognition for Australian historical newspapers." In: *Australasian Language Technology Association Workshop*, pp. 57–65. URL: https://aclanthology.org/U15-1007 (cit. on p. 65).

Kingma, D. and J. Ba (2014). "Adam: A method for stochastic optimization." In: *International Conference on Learning Representations (ICLR)*. URL: http://arxiv.org/abs/1412.6980 (cit. on p. 29).

Kitano, H. (2004). "Biological robustness." In: *Nature Reviews Genetics* 5.11, pp. 826–837. ISSN: 1471-0064. DOI: 10.1038/nrg1471 (cit. on p. 1).

Kleene, S. C. (1951). *Representation of events in nerve nets and finite automata*. Tech. rep. Rand Project Air Force Santa Monica, CA. URL: https://apps.dtic.mil/sti/pdfs/ADA596138.pdf (cit. on pp. 5, 111).

Klein, G., Y. Kim, Y. Deng, J. Senellart, and A. Rush (2017). "OpenNMT: Open-source toolkit for neural machine translation." In: *Annual Meeting of the Association for Computational Linguistics,System Demonstrations*. Association for Computational Linguistics, pp. 67–72. URL: https://aclanthology.org/P17-4012 (cit. on p. 83).

Kolak, O. and P. Resnik (2002). “OCR error correction using a noisy channel model.” In: *International Conference on Human Language Technology Research*. Morgan Kaufmann Publishers Inc., pp. 257–262. URL: https://dl.acm.org/doi/10.5555/1289189.1289208 (cit. on p. 74).

Kolak, O. and P. Resnik (2005). “OCR post-processing for low density languages.” In: *Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 867–874. URL: https://aclanthology.org/H05-1109 (cit. on p. 74).

Konya, I. (2013). “Adaptive methods for robust document image understanding.” PhD thesis. Rheinische Friedrich-Wilhelms-Universität Bonn. eprint: https://hdl.handle.net/20.500.11811/5655 (cit. on pp. 26, 113).

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). “ImageNet classification with deep convolutional neural networks.” In: *International Conference on Neural Information Processing Systems*. Vol. 1. Curran Associates Inc., pp. 1097–1105. DOI: 10.1145/3065386 (cit. on pp. 14, 48, 65).

Lafferty, J. D., A. McCallum, and F. C. N. Pereira (2001). “Conditional random fields: Probabilistic models for segmenting and labeling sequence data.” In: *International Conference on Machine Learning (ICML)*. Morgan Kaufmann Publishers Inc., pp. 282–289. ISBN: 1-55860-778-1. URL: http://dl.acm.org/citation.cfm?id=645530.655813 (cit. on p. 43).

Lage-Rupprecht, V., B. Schultz, J. Dick, M. Namysl, A. Zaliani, et al. (2022). “A hybrid approach unveils drug repurposing candidates targeting an Alzheimer pathophysiology mechanism.” In: *Patterns* 3.3, p. 100433. ISSN: 2666-3899. DOI: 10.1016/j.patter.2021.100433 (cit. on p. 7).

Lakshmi Narayan, P., A. Nagesh, and M. Surdeanu (2019). “Exploration of noise strategies in semi-supervised named entity classification.” In: *Joint Conference on Lexical and Computational Semantics (*SEM)*. Association for Computational Linguistics, pp. 186–191. DOI: 10.18653/v1/S19-1020 (cit. on p. 66).

Lample, G., M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer (2016). “Neural architectures for named entity recognition.” In: *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pp. 260–270. DOI: 10.18653/v1/N16-1030 (cit. on pp. 42, 52).

Lavirotte, S. and L. Pottier (1997). “Optical formula recognition.” In: *International Conference on Document Analysis and Recognition*. Vol. 1, pp. 357–361. DOI: 10.1109/ICDAR.1997.619871 (cit. on p. 2).

Le, D. S., G. R. Thoma, and H. Wechsler (1994). “Automated page orientation and skew angle detection for binary document images.” In: *Pattern Recognition*

27.10, pp. 1325–1344. ISSN: 0031-3203. DOI: 10.1016/0031-3203(94)90068-X (cit. on p. 2).

Le, H. and A. Borji (2017). *What are the receptive, effective receptive, and projective fields of neurons in convolutional neural networks?* DOI: 10.48550/ARXIV.1705.07049 (cit. on p. 14).

Lecun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998). "Gradient-based learning applied to document recognition." In: *Proceedings of the IEEE* 86.11, pp. 2278–2324. ISSN: 0018-9219. DOI: 10.1109/5.726791 (cit. on p. 14).

Lee, C.-Y. and S. Osindero (2016). "Recursive recurrent nets with attention modeling for OCR in the wild." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE Computer Society, pp. 2231–2239. DOI: 10.1109/CVPR.2016.245 (cit. on p. 19).

Lee, K., L. He, M. Lewis, and L. Zettlemoyer (2017). "End-to-end neural coreference resolution." In: *Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics, pp. 188–197. DOI: 10.18653/v1/D17-1018 (cit. on p. 2).

Levenshtein, V. I. (1966). "Binary codes capable of correcting deletions, insertions, and reversals." In: *Soviet Physics Doklady* 10.8. URL: https://nymity.ch/sybilhunting/pdf/Levenshtein1966a.pdf (cit. on pp. 30, 45, 76, 81, 111, 123).

Li, M., L. Cui, S. Huang, F. Wei, M. Zhou, and Z. Li (2020). "TableBank: Table benchmark for image-based table detection and recognition." In: *Language Resources and Evaluation Conference (LREC).* European Language Resources Association, pp. 1918–1925. ISBN: 979-10-95546-34-4. URL: https://aclanthology.org/2020.lrec-1.236 (cit. on pp. 133, 135, 138, 148, 162).

Li, M., T. Lv, L. Cui, Y. Lu, D. Florencio, C. Zhang, Z. Li, and F. Wei (2021). *TrOCR: Transformer-based optical character recognition with pre-trained models.* DOI: 10.48550/ARXIV.2109.10282 (cit. on p. 19).

Li, X., H. Xue, W. Chen, Y. Liu, Y. Feng, and Q. Liu (2018). "Improving the robustness of speech translation." In: DOI: 10.48550/ARXIV.1811.00728 (cit. on p. 65).

Li, Y., T. Cohn, and T. Baldwin (2016). "Learning robust representations of text." In: *Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics, pp. 1979–1985. DOI: 10.18653/v1/D16-1207 (cit. on p. 67).

Lichtarge, J., C. Alberti, S. Kumar, N. Shazeer, N. Parmar, and S. Tong (2019). "Corpora generation for grammatical error correction." In: *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers).* Association for Computational Linguistics, pp. 3291–3301. DOI: 10.18653/v1/N19-1333 (cit. on p. 75).

Lim, S., I. Kim, T. Kim, C. Kim, and S. Kim (2019). "Fast AutoAugment." In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., pp. 6665–6675. URL: http://papers.nips.cc/paper/8892-fast-autoaugment.pdf (cit. on p. 65).

Liu, L., Z. Wang, J. Shang, D. Yin, H. Ji, X. Ren, S. Wang, and J. Han (2019). *Raw-to-end name entity recognition in social media*. DOI: 10.48550/ARXIV.1908.05344 (cit. on p. 69).

Long, S., X. He, and C. Yao (2021). "Scene text detection and recognition: The deep learning era." In: *International Journal of Computer Vision* 129.1, pp. 161–184. DOI: 10.1007/s11263-020-01369-0 (cit. on p. 10).

Lopresti, D. (2009). "Optical character recognition errors and their effects on natural language processing." In: *International Journal on Document Analysis and Recognition (IJDAR)* 12.3, pp. 141–151. ISSN: 1433-2825. DOI: 10.1007/s10032-009-0094-8 (cit. on p. 65).

Lund, W. B., D. D. Walker, and E. K. Ringger (2011). "Progressive alignment and discriminative error correction for multiple OCR engines." In: *International Conference on Document Analysis and Recognition (ICDAR)*, pp. 764–768. DOI: 10.1109/ICDAR.2011.303 (cit. on p. 74).

Luo, W., Y. Li, R. Urtasun, and R. Zemel (2016). "Understanding the effective receptive field in deep convolutional neural networks." In: *International Conference on Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc., pp. 4905–4913. ISBN: 9781510838819. URL: https://proceedings.neurips.cc/paper/2016/file/c8067ad1937f728f51288b3eb986afaa-Paper.pdf (cit. on p. 14).

Luong, T., H. Pham, and C. D. Manning (2015). "Effective approaches to attention-based neural machine translation." In: *Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 1412–1421. DOI: 10.18653/v1/D15-1166 (cit. on p. 83).

Lyu, P., M. Liao, C. Yao, W. Wu, and X. Bai (2018). "Mask TextSpotter: An end-to-end trainable neural network for spotting text with arbitrary shapes." In: *Computer Vision – ECCV 2018*. Vol. 11218. Springer International Publishing, pp. 71–88. ISBN: 978-3-030-01264-9. DOI: 10.1007/978-3-030-01264-9_5 (cit. on p. 11).

Macdonald, E. and D. Barbosa (2020). "Neural relation extraction on Wikipedia tables for augmenting knowledge graphs." In: *ACM International Conference on Information and Knowledge Management*. Association for Computing Machinery, pp. 2133–2136. ISBN: 9781450368599. DOI: 10.1145/3340531.3412164 (cit. on p. 111).

Mamede, N., J. Baptista, and F. Dias (2016). “Automated anonymization of text documents.” In: *IEEE Congress on Evolutionary Computation (CEC)*. IEEE, pp. 1287–1294. DOI: 10.1109/CEC.2016.7743936 (cit. on p. 69).

Mikolov, T., E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin (2018). “Advances in pre-training distributed word representations.” In: *International Conference on Language Resources and Evaluation (LREC)*. European Language Resources Association (ELRA). URL: https://aclanthology.org/L18-1008 (cit. on pp. 51, 57).

Mikolov, T., I. Sutskever, K. Chen, G. Corrado, and J. Dean (2013). “Distributed representations of words and phrases and their compositionality.” In: *Advances in Neural Information Processing Systems*. Vol. 26. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf (cit. on p. 77).

Mikolov, T., K. Chen, G. Corrado, and J. Dean (2013). “Efficient estimation of word representations in vector space.” In: *International Conference on Learning Representations (ICLR), Workshop Track Proceedings*. URL: http://arxiv.org/abs/1301.3781 (cit. on p. 111).

Miller, D., S. Boisen, R. Schwartz, R. Stone, and R. Weischedel (2000). “Named entity extraction from noisy input: Speech and OCR.” In: *Applied Natural Language Processing Conference*. Association for Computational Linguistics, pp. 316–324. DOI: 10.3115/974147.974191 (cit. on pp. 3, 72).

Miyato, T., A. M. Dai, and I. J. Goodfellow (2017). “Adversarial training methods for semi-supervised text classification.” In: *International Conference on Learning Representations (ICLR)*. URL: https://openreview.net/forum?id=r1X3g2_xl (cit. on pp. 47, 68).

Mole, D., C. O’Neill, P. Hamilton, B. Olabi, V. Robinson, L. Williams, T. Diamond, M. El-Tanani, and F. Campbell (2011). “Expression of osteopontin coregulators in primary colorectal cancer and associated liver metastases.” In: *British Journal of Cancer* 104.6, pp. 1007–1012. DOI: 10.1038/bjc.2011.33 (cit. on p. 151).

Müller, M., A. Rios, and R. Sennrich (2020). “Domain robustness in neural machine translation.” In: *Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*. Association for Machine Translation in the Americas, pp. 151–164. URL: https://aclanthology.org/2020.amta-research.14 (cit. on p. 1).

Namysl, M., S. Behnke, and J. Köhler (2020). “NAT: Noise-aware training for robust neural sequence labeling.” In: *Annual Meeting of the Association for*

Computational Linguistics. Association for Computational Linguistics, pp. 1501–1517. DOI: 10.18653/v1/2020.acl-main.138 (cit. on pp. 6, 8, 39, 179).

Namysl, M., S. Behnke, and J. Köhler (2021). "Empirical error modeling improves robustness of noisy neural sequence labeling." In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP*. Association for Computational Linguistics, pp. 314–329. DOI: 10.18653/v1/2021.findings-acl.27 (cit. on pp. 7, 8, 39, 71, 180).

Namysl, M., A. Esser, S. Behnke, and J. Köhler (2022). "Flexible table recognition and semantic interpretation system." In: *International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. Vol. 4: VISAPP. INSTICC. SciTePress, pp. 27–37. ISBN: 978-989-758-555-5. DOI: 10.5220/0010767600003124 (cit. on pp. 7, 8, 103, 180).

Namysl, M. and I. Konya (2019). "Efficient, lexicon-free OCR using deep learning." In: *International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, pp. 295–301. DOI: 10.1109/ICDAR.2019.00055 (cit. on pp. 6, 9, 179).

Natarajan, P., S. Saleem, R. Prasad, E. MacRostie, and K. Subramanian (2008). "Multi-lingual offline handwriting recognition using hidden Markov models: A script-independent approach." In: *Arabic and Chinese Handwriting Recognition*. Springer. Springer Berlin Heidelberg, pp. 231–250. ISBN: 978-3-540-78199-8. DOI: 10.1007/978-3-540-78199-8_14 (cit. on p. 17).

Nayef, N., M. M. Luqman, S. Prum, S. Eskenazi, J. Chazalon, and J.-M. Ogier (2015). "SmartDoc-QA: A dataset for quality assessment of smartphone captured document images - single and multiple distortions." In: *International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1231–1235. DOI: 10.1109/ICDAR.2015.7333960 (cit. on p. 11).

Neudecker, C. (2016). "An open corpus for named entity recognition in historic newspapers." In: *International Conference on Language Resources and Evaluation (LREC)*. European Language Resources Association (ELRA), pp. 4348–4352. URL: https://aclanthology.org/L16-1689 (cit. on pp. 40, 65).

Ng, H. T., S. M. Wu, T. Briscoe, C. Hadiwinoto, R. H. Susanto, and C. Bryant (2014). "The CoNLL-2014 shared task on grammatical error correction." In: *Conference on Computational Natural Language Learning: Shared Task*. Association for Computational Linguistics, pp. 1–14. DOI: 10.3115/v1/W14-1701 (cit. on p. 74).

Ng, H. T., S. M. Wu, Y. Wu, C. Hadiwinoto, and J. Tetreault (2013). "The CoNLL-2013 shared task on grammatical error correction." In: *Conference on Computational Natural Language Learning: Shared Task*. Association for Computational Linguistics, pp. 1–12. URL: https://aclanthology.org/W13-3601 (cit. on p. 74).

Nurminen, A. (2013). “Algorithmic extraction of data in tables in PDF documents.” MA thesis. Tampere University of Technology. eprint: https://urn.fi/URN:NBN:fi:tty-201305231166 (cit. on pp. 110, 154).

Oro, E. and M. Ruffolo (2009). “PDF-TREX: An approach for recognizing and extracting tables from PDF documents.” In: *International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, pp. 906–910. DOI: 10.1109/ICDAR.2009.12 (cit. on pp. 107, 110).

Otsu, N. (1979). “A threshold selection method from gray-level histograms.” In: *IEEE Transactions on Systems, Man and Cybernetics* 9.1, pp. 62–66. DOI: 10.1109/TSMC.1979.4310076 (cit. on p. 114).

Packer, T. L., J. F. Lutes, A. P. Stewart, D. W. Embley, E. K. Ringger, K. D. Seppi, and L. S. Jensen (2010). “Extracting person names from diverse and noisy OCR text.” In: *Workshop on Analytics for Noisy Unstructured Text Data*. Association for Computing Machinery, pp. 19–26. ISBN: 9781450303767. DOI: 10.1145/1871840.1871845 (cit. on pp. 3, 65).

Paliwal, S. S., V. D, R. Rahul, M. Sharma, and L. Vig (2019). “TableNet: Deep learning model for end-to-end table detection and tabular data extraction from scanned document images.” In: *International Conference on Document Analysis and Recognition (ICDAR)*, pp. 128–133. DOI: 10.1109/ICDAR.2019.00029 (cit. on pp. 111, 154).

Papineni, K., S. Roukos, T. Ward, and W.-J. Zhu (2002). “BLEU: A method for automatic evaluation of machine translation.” In: *Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 311–318. DOI: 10.3115/1073083.1073135 (cit. on pp. 148, 161).

Parada, C., M. Dredze, and F. Jelinek (2011). “OOV sensitive named-entity recognition in speech.” In: *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pp. 2085–2088. DOI: 10.21437/Interspeech.2011-547 (cit. on p. 40).

Pascanu, R., T. Mikolov, and Y. Bengio (2013). “On the difficulty of training recurrent neural networks.” In: *International Conference on Machine learning (ICML)*. Vol. 28. JMLR.org, pp. 1310–1318. URL: http://proceedings.mlr.press/v28/pascanu13.pdf (cit. on p. 13).

Pawlik, M. and N. Augsten (2016). “Tree edit distance: Robust and memory-efficient.” In: *Information Systems* 56, pp. 157–173. ISSN: 0306-4379. DOI: 10.1016/j.is.2015.08.004 (cit. on pp. 148, 161).

Pennington, J., R. Socher, and C. Manning (2014). “GloVe: Global vectors for word representation.” In: *Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 1532–1543. DOI: 10.3115/v1/D14-1162 (cit. on pp. 51, 77).

Peters, M. E., M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer (2018). “Deep contextualized word representations.” In: *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers).* Association for Computational Linguistics, pp. 2227–2237. DOI: `10.18653/v1/N18-1202` (cit. on pp. 42, 52, 77).

Piktus, A., N. B. Edizel, P. Bojanowski, E. Grave, R. Ferreira, and F. Silvestri (2019). “Misspelling oblivious word embeddings.” In: *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, volume 1 (Long and Short Papers).* Association for Computational Linguistics, pp. 3226–3234. DOI: `10.18653/v1/N19-1326` (cit. on pp. 41, 53, 55, 56, 58, 59, 67, 82, 88).

Porter, T. and T. Duff (1984). “Compositing digital images.” In: *SIGGRAPH Computer Graphics* 18.3, pp. 253–259. ISSN: 0097-8930. DOI: `10.1145/964965.808606` (cit. on p. 23).

Prasad, D., A. Gadpal, K. Kapadni, M. Visave, and K. Sultanpure (2020). “CascadeTabNet: An approach for end to end table detection and structure recognition from image-based documents.” In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 2439–2447. DOI: `10.1109/CVPRW50498.2020.00294` (cit. on pp. 111, 132, 135, 138, 154).

Puigcerver, J. (2017). “Are multidimensional recurrent layers really necessary for handwritten text recognition?” In: *IAPR International Conference on Document Analysis and Recognition (ICDAR).* Vol. 01, pp. 67–72. DOI: `10.1109/ICDAR.2017.20` (cit. on p. 18).

Qiu, M. and J. Park (2019). “Artificial error generation with fluency filtering.” In: *Workshop on Innovative Use of NLP for Building Educational Applications.* Association for Computational Linguistics, pp. 87–91. DOI: `10.18653/v1/W19-4408` (cit. on p. 75).

Rabiner, L. R. (1989). “A tutorial on hidden Markov models and selected applications in speech recognition.” In: *Proceedings of the IEEE* 77.2, pp. 257–286. ISSN: 0018-9219. DOI: `10.1109/5.18626` (cit. on p. 18).

Raffel, C., N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu (2020). “Exploring the limits of transfer learning with a unified text-to-text transformer.” In: *Journal of Machine Learning Research* 21.140, pp. 1–67. URL: `http://jmlr.org/papers/v21/20-074.html` (cit. on p. 19).

Rashid, S. F., F. Shafait, and T. M. Breuel (2012). “Scanning neural network for text line recognition.” In: *IAPR International Workshop on Document Analysis Systems*, pp. 105–109. DOI: `10.1109/DAS.2012.77` (cit. on p. 17).

Rastan, R., H.-Y. Paik, and J. Shepherd (2015). “TEXUS: A task-based approach for table extraction and understanding.” In: *ACM Symposium on Document Engineering (DocEng)*, pp. 25–34. ISBN: 9781450333078. DOI: 10.1145/2682571.2797069 (cit. on pp. 110, 135).

Ratinov, L. and D. Roth (2009). “Design challenges and misconceptions in named entity recognition.” In: *Conference on Computational Natural Language Learning (CoNLL)*. Association for Computational Linguistics, pp. 147–155. URL: https://aclanthology.org/W09-1119 (cit. on p. 43).

Rawlinson, G. (2007). “The significance of letter position in word recognition.” In: *IEEE Aerospace and Electronic Systems Magazine* 22.1, pp. 26–27. ISSN: 0885-8985. DOI: 10.1109/MAES.2007.327521 (cit. on p. 41).

Rei, M., M. Felice, Z. Yuan, and T. Briscoe (2017). “Artificial error generation with machine translation and syntactic patterns.” In: *Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, pp. 287–292. DOI: 10.18653/v1/W17-5032 (cit. on pp. 75, 78).

Ren, S., K. He, R. Girshick, and J. Sun (2015). “Faster R-CNN: Towards real-time object detection with region proposal networks.” In: *Advances in Neural Information Processing Systems*. Vol. 28. Curran Associates, Inc. (cit. on p. 133).

Reza, M. M., S. S. Bukhari, M. Jenckel, and A. Dengel (2019). “Table localization and segmentation using GAN and CNN.” In: *International Conference on Document Analysis and Recognition Workshops (ICDARW)*. Vol. 5, pp. 152–157. DOI: 10.1109/ICDARW.2019.40097 (cit. on p. 111).

Reza Yousefi, M., M. R. Soheili, T. M. Breuel, and D. Stricker (2015). “A comparison of 1D and 2D LSTM architectures for the recognition of handwritten Arabic.” In: *The International Society for Optical Engineering (SPIE)* 9402. DOI: 10.1117/12.2075930 (cit. on p. 17).

Rifai, S., P. Vincent, X. Muller, X. Glorot, and Y. Bengio (2011). “Contractive auto-encoders: Explicit invariance during feature extraction.” In: *International Conference on Machine Learning (ICML)*. Omnipress, pp. 833–840. ISBN: 978-1-4503-0619-5. URL: http://dl.acm.org/citation.cfm?id=3104482.3104587 (cit. on p. 67).

Rigaud, C., A. Doucet, M. Coustaty, and J. Moreux (2019). “ICDAR 2019 competition on post-OCR text correction.” In: *International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1588–1593. DOI: 10.1109/ICDAR.2019.00255 (cit. on pp. 74, 78).

Ruiz, N., M. A. D. Gangi, N. Bertoldi, and M. Federico (2017). “Assessing the tolerance of neural machine translation systems against speech recognition errors.” In: *Annual Conference of the International Speech Communication*

Association (INTERSPEECH), pp. 2635–2639. DOI: 10.21437/Interspeech.2017-1690 (cit. on p. 65).

Sabir, E., S. Rawls, and P. Natarajan (2017). "Implicit language model in LSTM for OCR." In: *IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 07, pp. 27–31. DOI: 10.1109/ICDAR.2017.361 (cit. on p. 30).

Sánchez, J. A., V. Romero, A. H. Toselli, M. Villegas, and E. Vidal (2017). "ICDAR2017 competition on handwritten text recognition on the READ dataset." In: *IAPR International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 01, pp. 1383–1388. DOI: 10.1109/ICDAR.2017.226 (cit. on p. 11).

Sauvola, J., T. Seppanen, S. Haapakoski, and M. Pietikainen (1997). "Adaptive document binarization." In: *International Conference on Document Analysis and Recognition*. Vol. 1, pp. 147–152. DOI: 10.1109/ICDAR.1997.619831 (cit. on p. 2).

Schmaltz, A., Y. Kim, A. Rush, and S. Shieber (2017). "Adapting sequence models for sentence correction." In: *Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 2807–2813. DOI: 10.18653/v1/D17-1298 (cit. on p. 74).

Schnober, C., S. Eger, E.-L. Do Dinh, and I. Gurevych (2016). "Still not there? Comparing traditional sequence-to-sequence models to encoder-decoder neural networks on monotone string translation tasks." In: *International Conference on Computational Linguistics (COLING): Technical Papers*. The COLING 2016 Organizing Committee, pp. 1703–1714. URL: https://aclanthology.org/C16-1160 (cit. on pp. 74, 78).

Schreiber, S., S. Agne, I. Wolf, A. Dengel, and S. Ahmed (2017). "DeepDeSRT: Deep learning for detection and structure recognition of tables in document images." In: *International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 01, pp. 1162–1167. DOI: 10.1109/ICDAR.2017.192 (cit. on p. 110).

Schulz, S. and J. Kuhn (2017). "Multi-modular domain-tailored OCR post-correction." In: *Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 2716–2726. DOI: 10.18653/v1/D17-1288 (cit. on p. 74).

Schuster, M. and K. K. Paliwal (1997). "Bidirectional recurrent neural networks." In: *IEEE Transactions on Signal Processing* 45.11, pp. 2673–2681. ISSN: 1053-587X. DOI: 10.1109/78.650093 (cit. on p. 13).

See, A., P. J. Liu, and C. D. Manning (2017). "Get to the point: Summarization with pointer-generator networks." In: *Annual Meeting of the Association for*

Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, pp. 1073–1083. DOI: 10.18653/v1/P17-1099 (cit. on p. 83).

Sennrich, R., B. Haddow, and A. Birch (2016). "Improving neural machine translation models with monolingual data." In: *Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, pp. 86–96. DOI: 10.18653/v1/P16-1009 (cit. on pp. 68, 75).

Shi, B., X. Bai, and C. Yao (2017). "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.11, pp. 2298–2304. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2016.2646371 (cit. on pp. 18, 20).

Shigarov, A., A. Altaev, A. Mikhailov, V. Paramonov, and E. Cherkashin (2018). "TabbyPDF: Web-based system for PDF table extraction." In: *Information and Software Technologies*. Springer, pp. 257–269. ISBN: 978-3-319-99972-2. DOI: 10.1007/978-3-319-99972-2_20 (cit. on pp. 107, 110, 135, 154).

Silva, A. C. e. (2011). "Metrics for evaluating performance in document analysis: Application to tables." In: *International Journal on Document Analysis and Recognition (IJDAR)* 14.1, pp. 101–109. ISSN: 1433-2833. DOI: 10.1007/s10032-010-0144-2 (cit. on p. 147).

Silva, A. C. e., A. M. Jorge, and L. Torgo (2005). "Design of an end-to-end method to extract information from tables." In: *International Journal of Document Analysis and Recognition (IJDAR)* 8, pp. 144–171. DOI: 10.1007/s10032-005-0001-x (cit. on p. 109).

Silveira, N., T. Dozat, M.-C. de Marneffe, S. Bowman, M. Connor, J. Bauer, and C. Manning (2014). "A gold standard dependency corpus for English." In: *International Conference on Language Resources and Evaluation (LREC)*. European Language Resources Association (ELRA), pp. 2897–2904. URL: http://www.lrec-conf.org/proceedings/lrec2014/pdf/1089_Paper.pdf (cit. on pp. 51, 86).

Simard, P. Y., D. Steinkraus, and J. C. Platt (2003). "Best practices for convolutional neural networks applied to visual document analysis." In: *International Conference on Document Analysis and Recognition (ICDAR)*, pp. 958–963. DOI: 10.1109/ICDAR.2003.1227801 (cit. on p. 23).

Simonyan, K. and A. Zisserman (2014). *Very deep convolutional networks for large-scale image recognition*. DOI: 10.48550/ARXIV.1409.1556 (cit. on p. 15).

Smith, R. (2011). "Limits on the application of frequency-based language models to OCR." In: *International Conference on Document Analysis and Recognition (ICDAR)*, pp. 538–542. DOI: 10.1109/ICDAR.2011.114 (cit. on pp. 11, 16).

Smith, R. (2007). "An overview of the Tesseract OCR engine." In: *International Conference on Document Analysis and Recognition (ICDAR)*. Vol. 2, pp. 629–633. DOI: 10.1109/ICDAR.2007.4376991 (cit. on pp. 27, 30, 84, 114).

Sperber, M., J. Niehues, and A. Waibel (2017). "Toward robust neural machine translation for noisy input sequences." In: *The International Workshop on Spoken Language Translation (IWSLT)*. URL: http://workshop2017.iwslt.org/downloads/P04-Paper.pdf (cit. on pp. 42, 48, 66).

Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov (2014). "Dropout: A simple way to prevent neural networks from overfitting." In: *Journal of Machine Learning Research* 15, pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html (cit. on pp. 30, 48, 167).

Sutskever, I., O. Vinyals, and Q. V. Le (2014). "Sequence to sequence learning with neural networks." In: *Advances in Neural Information Processing Systems*. Vol. 27. Curran Associates, Inc., pp. 3104–3112. URL: https://proceedings.neurips.cc/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf (cit. on p. 5).

Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich (2015). "Going deeper with convolutions." In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9. DOI: 10.1109/CVPR.2015.7298594 (cit. on p. 15).

Szegedy, C., W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus (2014). "Intriguing properties of neural networks." In: *International Conference on Learning Representations (ICLR)*. URL: https://openreview.net/forum?id=kklr_MTHMRQjG (cit. on p. 68).

Tengli, A., Y. Yang, and N. L. Ma (2004). "Learning table extraction from examples." In: *International Conference on Computational Linguistics*. COLING, pp. 987–993. URL: https://aclanthology.org/C04-1142 (cit. on p. 2).

Tjong Kim Sang, E. F. and S. Buchholz (2000). "Introduction to the CoNLL-2000 shared task chunking." In: *Conference on Computational Natural Language Learning and the Learning Language in Logic Workshop*. URL: https://aclanthology.org/W00-0726 (cit. on p. 51).

Tjong Kim Sang, E. F. and F. De Meulder (2003). "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition." In: *Conference on Natural Language Learning at HLT-NAACL*, pp. 142–147. URL: https://aclanthology.org/W03-0419 (cit. on pp. 2, 44, 47, 50, 86).

Toutanova, K. and R. C. Moore (2002). "Pronunciation modeling for improved spelling correction." In: *Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 144–151. DOI: 10.3115/1073083.1073109 (cit. on p. 74).

Treue, S. and S. Katzner (2009). “Visual attention.” In: *Encyclopedia of Neuroscience*. Academic Press, pp. 243–250. ISBN: 978-0-08-045046-9. DOI: 10.1016/B978-008045046-9.00242-4 (cit. on p. 19).

Tsvetkov, Y., F. Metze, and C. Dyer (2014). “Augmenting translation models with simulated acoustic confusions for improved spoken language translation.” In: *Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 616–625. DOI: 10.3115/v1/E14-1065 (cit. on p. 65).

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin (2017). “Attention is all you need.” In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf (cit. on p. 19).

Voigtlaender, P., P. Doetsch, and H. Ney (2016). “Handwriting recognition with large multidimensional long short-term memory recurrent neural networks.” In: *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 228–233. DOI: 10.1109/ICFHR.2016.0052 (cit. on p. 17).

Wang, A., A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman (2018). “GLUE: A multi-task benchmark and analysis platform for natural language understanding.” In: *EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Association for Computational Linguistics, pp. 353–355. DOI: 10.18653/v1/W18-5446 (cit. on p. 72).

Wang, D., Y. Song, J. Li, J. Han, and H. Zhang (2018). “A hybrid approach to automatic corpus generation for Chinese spelling check.” In: *Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 2517–2527. DOI: 10.18653/v1/D18-1273 (cit. on p. 80).

Wang, J., K. Sun, T. Cheng, B. Jiang, C. Deng, et al. (2021). “Deep high-resolution representation learning for visual recognition.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.10, pp. 3349–3364. DOI: 10.1109/TPAMI.2020.2983686 (cit. on p. 126).

Wang, K., B. Babenko, and S. Belongie (2011). “End-to-end scene text recognition.” In: *International Conference on Computer Vision*, pp. 1457–1464. DOI: 10.1109/ICCV.2011.6126402 (cit. on p. 11).

Webster, J. J. and C. Kit (1992). “Tokenization as the initial phase in NLP.” In: *Conference on Computational Linguistics - Volume 4*. Association for Computational Linguistics, pp. 1106–1110. DOI: 10.3115/992424.992434 (cit. on p. 2).

Wemhoener, D., I. Z. Yalniz, and R. Manmatha (2013). “Creating an improved version using noisy OCR from multiple editions.” In: *International Conference* *on Document Analysis and Recognition*, pp. 160–164. DOI: 10.1109/ICDAR.2013.39 (cit. on p. 74).

Werbos, P. J. (1990). “Backpropagation through time: What it does and how to do it.” In: *Proceedings of the IEEE* 78.10, pp. 1550–1560. ISSN: 0018-9219. DOI: 10.1109/5.58337 (cit. on p. 12).

Wright, P. (1980). “The comprehension of tabulated information: Some similarities between reading prose and reading tables.” In: *NSPI Journal* 19.8, pp. 25–29. DOI: https://doi.org/10.1002/pfi.4180190810 (cit. on p. 104).

Xia, J., H. Hu, W. Xue, X. S. Wang, and S. Wu (2018). “The discovery of novel HDAC3 inhibitors via virtual screening and in vitro bioassay.” In: *Journal of Enzyme Inhibition and Medicinal Chemistry* 33.1, pp. 525–535. DOI: 10.1080/14756366.2018.1437156. eprint: https://doi.org/10.1080/14756366.2018.1437156 (cit. on p. 125).

Xie, Q., Z. Dai, E. Hovy, T. Luong, and Q. Le (2020). “Unsupervised data augmentation for consistency training.” In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., pp. 6256–6268. URL: https://proceedings.neurips.cc/paper/2020/file/44feb0096faa8326192570788b38c1d1-Paper.pdf (cit. on p. 67).

Xie, S., R. Girshick, P. Dollar, Z. Tu, and K. He (2017). “Aggregated residual transformations for deep neural networks.” In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on pp. 126, 133).

Xie, Z., G. Genthial, S. Xie, A. Ng, and D. Jurafsky (2018). “Noising and denoising natural language: Diverse backtranslation for grammar correction.” In: *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, pp. 619–628. DOI: 10.18653/v1/N18-1057 (cit. on pp. 75, 78).

Xie, Z., S. I. Wang, J. Li, D. Lévy, A. Nie, D. Jurafsky, and A. Y. Ng (2017). “Data noising as smoothing in neural network language models.” In: *International Conference on Learning Representations (ICLR), Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=H1VyHY9gg (cit. on p. 81).

Xu, Z., Y. Yang, and A. G. Hauptmann (2015). “A discriminative CNN video representation for event detection.” In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1798–1807. DOI: 10.1109/CVPR.2015.7298789 (cit. on p. 14).

Yan, C. and Y. He (2018). “Synthesizing type-detection logic for rich semantic data types using open-source code.” In: *International Conference on Management* *of Data (SIGMOD)*. Association for Computing Machinery, pp. 35–50. ISBN: 9781450347037. DOI: 10.1145/3183713.3196888 (cit. on p. 111).

Yasunaga, M., J. Kasai, and D. Radev (2018). "Robust multilingual part-of-speech tagging via adversarial training." In: *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, pp. 976–986. DOI: 10.18653/v1/N18-1089 (cit. on pp. 47, 68).

Yousefi, M. R., M. R. Soheili, T. M. Breuel, E. Kabir, and D. Stricker (2015). "Binarization-free OCR for historical documents using LSTM networks." In: *International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1121–1125. DOI: 10.1109/ICDAR.2015.7333935 (cit. on p. 17).

Yu, T., C.-S. Wu, X. V. Lin, bailin wang, Y. C. Tan, X. Yang, D. Radev, richard socher, and C. Xiong (2021). "GraPPa: Grammar-augmented pre-training for table semantic parsing." In: *International Conference on Learning Representations (ICLR)*. URL: https://openreview.net/forum?id=kyaIeYj4zZ (cit. on p. 111).

Yuan, Z. and T. Briscoe (2016). "Grammatical error correction using neural machine translation." In: *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pp. 380–386. DOI: 10.18653/v1/N16-1042 (cit. on p. 74).

Zhang, D., M. Hulsebos, Y. Suhara, Ç. Demiralp, J. Li, and W.-C. Tan (2020). "Sato: Contextual semantic type detection in tables." In: *VLDB Endowment* 13.12, pp. 1835–1848. ISSN: 2150-8097. DOI: 10.14778/3407790.3407793 (cit. on p. 112).

Zhang, H., H. Zhou, N. Miao, and L. Li (2019). "Generating fluent adversarial examples for natural languages." In: *Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 5564–5569. DOI: 10.18653/v1/P19-1559 (cit. on p. 68).

Zhang, Q., M. Chen, and L. Liu (2017). "A review on entity relation extraction." In: *International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, pp. 178–183. DOI: 10.1109/ICMCCE.2017.14 (cit. on p. 2).

Zhang, Z., X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu (2019). "ERNIE: Enhanced language representation with informative entities." In: *Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 1441–1451. DOI: 10.18653/v1/P19-1139 (cit. on p. 72).

Zheng, S., Y. Song, T. Leung, and I. J. Goodfellow (2016). "Improving the robustness of deep neural networks via stability training." In: *IEEE Conference* *on Computer Vision and Pattern Recognition (CVPR)*, pp. 4480–4488. DOI: 10.1109/CVPR.2016.485 (cit. on pp. 42, 48, 67).

Zheng, Y., C. Liu, X. Ding, and S. Pan (2001). "Form frame line detection with directional single-connected chain." In: *International Conference on Document Analysis and Recognition*, pp. 699–703. DOI: 10.1109/ICDAR.2001.953880 (cit. on p. 114).

Zhong, X., E. ShafieiBavani, and A. Jimeno Yepes (2020). "Image-based table recognition: Data, model, and evaluation." In: *Computer Vision – ECCV 2020*. Springer International Publishing, pp. 564–580. ISBN: 978-3-030-58589-1. DOI: 10.1007/978-3-030-58589-1_34 (cit. on pp. 146, 148).

Zhu, X., W. Su, L. Lu, B. Li, X. Wang, and J. Dai (2021). "Deformable DETR: Deformable transformers for end-to-end object detection." In: *International Conference on Learning Representations (ICLR)*. URL: https://openreview.net/forum?id=gZ9hCDWe6ke (cit. on p. 19).