Improving Optical Character Recognition for Endangered Languages

Shruti Rijhwani

CMU-LTI-22-016

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213

Thesis Committee:

Graham Neubig (Chair), Carnegie Mellon University

Alan Black, Carnegie Mellon University

Taylor Berg-Kirkpatrick, UC San Diego/Carnegie Mellon University

Antonios Anastasopoulos, George Mason University

Daisy Rosenblum, University of British Columbia

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Language and Information Technologies

Abstract

Much of the text data that exists in many languages is locked away in non-digitized books and documents. This is particularly true in the case of most endangered languages, where little to no machine-readable text is available, but printed materials such as cultural texts, educational books, and notes from linguistic documentation frequently exist. Extracting text from these materials to a machine-readable format is useful for a multitude of reasons. It can aid endangered language preservation and accessibility efforts by archiving the texts and making them searchable for language learners and speakers, as well as enable the development of natural language processing systems for endangered languages.

Optical character recognition (OCR) is typically used to extract text from such documents, but state-of-the-art OCR systems need large amounts of text data and transcribed images to train highly-performant models. These resources are often unavailable for endangered languages and because OCR models are not designed to work well in low-resource scenarios, transcriptions of endangered language documents are far less accurate than higher-resourced counterparts.

In this thesis, we address the task of improving OCR in order to produce high-quality transcriptions of documents that contain text in endangered languages. We use the technique of OCR post-correction, where the goal is to correct errors in existing OCR outputs to increase accuracy. We propose a suite of methods that are tailored to learning from small amounts of data, and empirically show significantly reduction in error rates than existing OCR systems in low-resource settings.

We first present a benchmark dataset for the task of OCR on endangered language texts, containing transcriptions of printed documents in four critically endangered languages, and extensively analyze the shortcomings of existing methods on this dataset, finding that there is considerable room for improvement. Then, we present two models for fixing recognition errors in OCR outputs, targeted to data-scarce settings: (1) a neural OCR post-correction method that leverages high-resource translations and structural biases to train a better-performing model; and (2) a semi-supervised technique that efficiently uses unlabeled scanned images (which are easier to obtain than manually annotated documents) for learning a post-correction model by combining self-training with automatically derived lexica.

Additionally, we investigate the real-world impact our proposed models could have on endangered language revitalization by conducting a comprehensive case study on the Kwak'wala language. The case study includes a human-centered evaluation that quantitatively analyzes the utility of post-correction in reducing the manual effort needed for language documentation tasks. Further, to make state-of-the-art OCR technologies (including our post-correction method) more accessible to users who may not have a technical background, we develop a web application that abstracts away software and scripting details and allows users to easily experiment with a variety of OCR tools and train models on new languages.

We make the software to use the post-correction models proposed in this thesis publicly available, with the hope of enabling model development for new languages and orthographies, and facilitating improvements in text recognition pipelines for low-resource and endangered languages at a global scale.

Acknowledgments

This thesis would not have been possible without the continuous support and encouragement I received from my mentors, collaborators, friends, and family. I am very grateful to my advisor, Graham Neubig, for teaching me practically everything about the research process and helping me grow into a well-rounded researcher. I am particularly thankful that Graham always supported my interest in building language technologies that have real-world impact and working with downstream users of our research (without which this thesis would look very different!). I would also like to thank Antonios Anastasopoulos for the brainstorming sessions that framed my research agenda for this thesis, for the continuous collaboration and feedback on my work over nearly three years, and for being on my thesis committee. Additionally, I am fortunate to have extensively collaborated with Daisy Rosenblum who, apart from serving on my thesis committee, taught me so much about the Kwak'wala language and community, language documentation and revitalization processes, and developing technology for indigenous and endangered languages. I am also grateful to Alan Black for the incredibly helpful discussions on research ideas and future directions for my thesis, and to Taylor Berg-Kirkpatrick for mentoring me on an independent study where I first explored OCR research and for the encouraging feedback as a thesis committee member. Further, I would like to note my gratitude to Jaime Carbonell, my co-advisor in the early years of my Ph.D. program – Jaime's advice through the LORELEI project and beyond helped shape my research interest in NLP for under-resourced languages.

I am immensely thankful to the endangered language communities for sharing their data and documents for our research as well as their feedback on the technologies presented in this thesis. I worked closely with Michayla King and Jaymyn LaVallee, heritage learners of the Kwak'wala language, whose tremendous efforts in data annotation were instrumental in developing the OCR pipeline for Kwak'wala.

Beyond the research presented in this thesis, I had the opportunity to work on a diverse set of projects during my time as a Ph.D. student – I am extremely grateful to all my research collaborators at CMU, from whom I learned about a broad range of research areas and picked up new technical skills. These include Shuyan Zhou, Lindia Tjuatja, Zaid Sheikh, Adithya Pratapa, Rosaline Su, the CMU LORELEI team, and the Print and Probability team. I am also thankful to Daniel

Preoțiuc-Pietro, Jing Wang, and Anju Kambadur, who mentored me through two exciting and rewarding internships at Bloomberg L.P.

I am very appreciative to have been part of the NeuLab group with brilliant researchers who I learned so much from. I am fortunate to take away fond memories of group dinners, happy hours, and barbecues. My six years at CMU wouldn't have been nearly as enjoyable without the support of my friends – thank you to Shuyan Zhou, Shruti Palaskar, Aakanksha Naik, Khyathi Chandu, Chirag Nagpal, Koushik Bhavani, Aditya Vamsikrishna, Dhruv Shah, Gayatri Bhat, Dheeraj Rajagopal, Vidhisha Balachandran, Ramon Sanabria, Sai Krishna, Alex Wilf, Patrick Fernandes, Cindy Wang, Chunting Zhou, Lucio Dery, Aditi Chalisgaonkar, Sunil Nayak, "Stefman" (and many more...) for all the dinners, hikes, road trips, and foosball matches. I want to express immense gratitude to my "Pittsburgh family" – Siddharth Dalmia, Rajat Kulshreshtha, Abhilasha Ravichander, Maria Ryskina, Samridhi Shree, Evangelia Spiliopoulou, and Paul Michel – I never imagined I would make the closest friends I've ever had at the LTI and I am very lucky to have met all of you. Thank you so much for all the good times over the years. I also thank Stacey Young, our wonderful program coordinator, for helping me navigate the administrative logistics throughout my time at CMU.

My family has been a constant source of strength and encouragement at every step of my Ph.D. journey – my parents and my sister have always been there for me even though we are separated by continents. Finally, words cannot express how grateful I am to my partner, Deepak Gopinath, whose unconditional love, support, and confidence in my abilities make me believe I can achieve anything.

Contents

Chapter 1

Introduction

Endangered languages are languages at risk of falling out of use because the population of first-language speakers is small and acquisition among younger speakers is limited. UNESCO classifies nearly 40% of the world's 7,000 living languages as endangered (Moseley, 2010). Based on the degree of intergenerational language transmission and other factors, these range from vulnerable languages where most children speak the language, but are restricted to certain domains, to critically endangered languages where the youngest speakers are grandparents and older, and they speak the language partially and infrequently.

The loss of a language has significant consequences: language is often an essential part of a community's cultural identity and heritage (Krauss, 1992; Craig, 1992) and, additionally, a considerable amount of knowledge is linguistically unique (i.e., it is only known to a single language and the community speaking that language). The history of a community and the cultural, spiritual, and scientific knowledge of its people is passed down through its language (Harrison, 2008; Cámara-Leret and Bascompte, 2021) and such knowledge may disappear as languages fall out of use (Hale et al., 1992). There are, thus, several ongoing efforts to preserve and revitalize endangered languages through language learning and education programs within the communities that speak these languages (Hinton, 2001) as well as documentation of endangered languages through the collection of speech recordings, word lists, and other materials.

The processes involved in language documentation, preservation, and revitalization frequently lead to the creation of several types of documents that contain text in endangered languages. For instance, local publishing houses print books containing folk tales, poetry, and other cultural texts, and language education programs often develop reference dictionaries and textbooks (Grenoble and Whaley, 2005). Additionally, the linguistic documentation process serves to develop a comprehensive record of the language (Himmelmann, 1998), resulting in various texts such as speech transcriptions, vocabulary lists, interlinear glosses, and grammar rules. However, even though a substantial number of such documents have been created for endangered languages around the globe, the vast majority are not widely accessible because they exist only as scanned images of printed books and handwritten notes. Beyond accessibility, converting these texts into a machine-readable format has several other benefits. Digitization can support language preservation efforts by archiving the texts and making them searchable for language learners and speakers, and the texts can also provide data to enable the development of natural language processing systems for endangered languages. These applications are discussed in more detail later in this chapter.

Optical character recognition (OCR) methods are typically used for extracting text from printed materials, where the characters in scanned images of the documents are recognized to produce transcriptions of the text. Both supervised and unsupervised learning techniques have been developed for training OCR models. In recent work, most supervised models are based on neural networks, where convolutional neural networks (CNN) are used to encode the input image (He et al., 2016; Shi et al., 2016; Gao et al., 2017) and Connectionist Temporal Classification (CTC) or an RNN-based attention mechanism is used for decoding the transcription (Chen et al., 2021; Long et al., 2021). These models generally require hundreds of thousands of transcribed images for supervised learning (Du et al., 2020). On the other hand, the majority of unsupervised OCR methods rely on large text corpora in the target language, which are used to generate synthetic images to train a neural model (Martínek et al., 2019; Vögtlin et al., 2021) or to build a language model that conditions the learning of character shapes to be recognized (Berg-Kirkpatrick et al., 2013).

However, applying these methods to obtain accurate transcriptions for documents that contain endangered language texts is challenging: the large number of transcribed images necessary for supervised training of OCR models is unavailable for most endangered languages. Additionally, the performance of unsupervised OCR systems is severely limited by the very small amounts of text data that is typically available in endangered languages (Littell et al., 2018; Rangel, 2019). Consequently, general-purpose OCR tools, such as Google Vision (Fujii, 2018) and Tesseract (Smith, 2007), are only trained on data from higher-resourced languages and do not provide off-the-shelf models for any endangered languages.

To overcome these challenges, this thesis presents a suite of methods designed to improve OCR in very low-resource settings. We first present the creation of a benchmark dataset for OCR on documents that contain text in four critically endangered languages (Ainu, Griko, Kwak'wala, and Yakkha) and extensively analyze the shortcomings of existing models on these

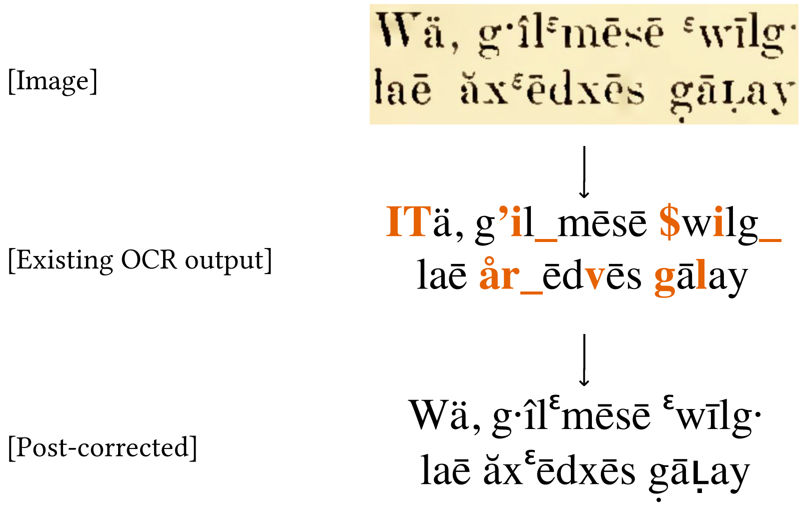


Figure 1.1: OCR post-correction on a scanned document that contains text in the endangered language Kwak’wala. The goal of post-correction is to fix the **recognition errors** in the output of an existing OCR system.

documents, demonstrating that they are not robust to low-resourced languages. Then, we empirically show how the quality of transcriptions from these existing OCR systems can be significantly improved by fixing recognition errors with automatic OCR post-correction. An example of the post-correction process is in Figure 1.1. We present two post-correction methods that are tailored to ease learning in data-scarce settings: (1) a neural post-correction model that uses structural biases and pretraining to reduce the reliance on manually labeled data for training; and (2) a semi-supervised learning technique that efficiently uses unlabeled raw images to improve the post-correction performance. We also conduct a study to evaluate the impact of our models on downstream use cases. The case study focuses on Kwak’wala, an endangered language spoken in North America, and includes a user evaluation that quantitatively establishes the utility of improved OCR in reducing the manual effort needed to produce accurate transcriptions of the texts. Finally, we present the development of a web application as a no-code interface between users and the post-correction software and other OCR tools, enabling much easier access to these technologies.

The high-quality transcriptions of endangered language documents that can be obtained with the methods presented in this thesis have a myriad of potential applications that can aid language documentation, education, and revitalization efforts:

- **Accessibility and search**: Having machine-readable transcriptions improves the accessibility of printed documents by archiving the texts and making them searchable, which benefits learners, speakers, and researchers of the language. As an example, having machine-readable reference texts such as dictionaries and grammars facilitates the creation of web and mobile interfaces for searching through these texts, making them more accessible and easier to use for language learners and educators.

- **Orthography conversion**: Many endangered languages have been written in different orthographies across geographies and generations, and historical documents in these languages may not be written in a modern orthography (Littell et al., 2018). Digitizing the text contained in these documents enables automatic transliteration from a legacy (or technical) orthography to a community-preferred orthography. For example, several decades ago, an orthography developed by anthropologist Franz Boas for the Kwak'wala language (Boas, 1900) was used to write an extensive documentation of the language and its speakers (Boas, 1921). However, these documents exist only as scanned images and are minimally accessible because the Boas orthography is hard to read and is radically different from modern orthographies. Converting the Boas publications to a machine-readable form facilitates automatic transliteration to different Kwak'wala writing systems, allowing community members to access the texts in their preferred orthography.

- **Development of NLP systems**: Most endangered languages are under-represented in natural language processing technologies, primarily because there is little to no data available for training and evaluation (Joshi et al., 2020). Machine-readable documents in endangered languages can provide text data for developing NLP systems in these languages. Apart from text in the endangered language itself, accurate OCR on documents like dictionaries and interlinear glosses will also create several other types of data useful for NLP (e.g., syntactic and morphological information, translations to higher-resourced languages, and bilingual lexicons).

  NLP systems for endangered languages can be used to support language documentation and revitalization. For instance, the language documentation process often involves manual transcription and glossing of speech recordings (Rangel, 2019), and language technologies such as automatic speech recognition and morphosyntactic analysis can significantly reduce the time and manual effort required to produce complete transcriptions (Cruz and Waring, 2019; Anastasopoulos, 2019; Anastasopoulos et al., 2020). Moreover, language education and revitalization efforts can leverage technologies such as automatic spell checking, text-to-speech, and predictive keyboards (Littell et al., 2018).

1.1 Thesis Overview

This thesis presents an evaluation dataset and methods for improving optical character recognition in very low-resource scenarios focusing on the application of these methods to endangered language texts. Additionally, the thesis analyzes the proposed technique's practical impact on language revitalization with a user study as well as develops a web interface for better access to OCR technologies.[1] An overview of the material presented in this thesis is below:

- Chapter 2 presents the creation of a benchmark dataset for evaluating the performance of OCR and OCR post-correction methods on endangered languages. The dataset contains manually transcribed pages from documents in four critically endangered languages – Ainu, Griko, Kwak'wala, and Yakkha. Further, this chapter presents an analysis of the performance of existing OCR systems on endangered languages, demonstrating that there is significant room for improvement in OCR transcription accuracy on the documents in our dataset.
- Chapter 3 presents the task of OCR post-correction, the goal of which is to improve the outputs of an existing OCR system. Existing methods for OCR post-correction are based on neural encoder-decoder models and rely on significant resources for training. In this chapter, we develop a method that compounds on previous neural models for OCR post-correction, adapting them to the under-resourced setting by introducing structural biases into the model and by using a multi-source sequence-to-sequence framework to incorporate information from the translations (often in a high-resource language) that commonly appear in endangered language texts. The proposed post-correction model substantially improves recognition error rates on the documents in our evaluation dataset as compared to existing OCR and OCR post-correction methods.
- While the post-correction model presented in Chapter 3 leads to improvement in OCR performance over previous work on the benchmark dataset, it is reliant on manually curated post-correction data, which is relatively scarce compared to the non-annotated raw images that need to be digitized. To efficiently utilize these unlabeled images, Chapter 4 presents a semi-supervised learning method for OCR post-correction through the use of self-training, a technique where the model is iteratively trained on its own outputs. In addition, to enforce consistency in the predicted vocabulary, we introduce a lexically-aware decoding method that augments the neural post-correction model with a word-by-word

[1] All code and data are publicly available at `https://shrutirij.github.io/ocr-el/`.

dictionary constructed from the predictions on the unlabeled images, implemented using weighted finite-state automata (WFSA) for efficient and effective decoding.

- The empirical reduction in character and word error rates obtained by the post-correction methods proposed in Chapter 3 and Chapter 4 is useful in comparing performance on low-resource datasets. However, our broader goal is for these methods to have an impact on downstream tasks that are important to endangered language speakers and researchers. In Chapter 5, we evaluate the potential impact of improved OCR on revitalization programs with a comprehensive case study on the Kwak'wala language. With a user evaluation, we demonstrate a significant reduction in the time needed for manual transcription of printed documents, a task that is frequently required in language documentation and preservation efforts. Additionally, we convert and publicly release several hundred pages of scanned cultural and linguistic documentation into a machine-readable format along with their automatic transliteration into different Kwak'wala orthographies that are preferred by community members.
- Most OCR and post-correction technologies are currently available only as command line tools or software APIs. The target audience for the methods developed in this thesis, such as endangered language educators and researchers, may not have the technical expertise necessary to access these technologies. In Chapter 6, we present a prototype visual interface for user-friendly access to OCR models, allowing no-code training and inference for OCR on new datasets and languages. The interface is designed as a web application and it supports various functionalities, including inference with off-the-shelf OCR systems as well as training and inference with our proposed post-correction models.

Chapter 2

Benchmark Dataset

In this chapter, we present the creation of a dataset for evaluating OCR performance on endangered language documents. The dataset contains manually transcribed documents in four critically endangered languages – Ainu, Griko, Kwak'wala, and Yakkha. We also evaluate the performance of existing supervised and unsupervised OCR methods on our dataset, extensively analyzing the successes and the shortcomings of these techniques. The material presented in this chapter has also been published in Rijhwani et al. (2020) and Rijhwani et al. (2021).

2.1 Endangered Language Documents

As briefly discussed in Chapter 1, the vast majority of documents that contain text in endangered languages exist only as non-digitized forms (e.g., printed books and handwritten documents). While a large fraction of these documents is still inaccessible on the web, many of them have been scanned and converted into images by linguists, endangered language communities, archivists, and librarians, among other entities. The scanned images of these documents are often publicly available through online archives.

To estimate how many such scanned documents with endangered language text exist, we explored some of these online archives. We first looked at the Internet Archive,[1] a general-purpose archive of web content. The Internet Archive contains millions of scanned books, most of which are labeled with the language of their content. On analyzing the language labels, we found 11,674 books containing text in languages classified as "endangered" by UNESCO (as of October 2020). Next, we looked at archives that preserve and catalog linguistic materials.

[1] https://archive.org/

We found that archives that specifically cater to Indigenous and endangered languages contain thousands of scanned documents, the majority of which come from linguistic documentation of these languages — the Archive of the Indigenous Languages of Latin America (AILLA)[2] contained ≈10,000 such documents and the Endangered Languages Archive (ELAR)[3] had ≈7,000.

Translations in a High-Resource Language We also observe that documents with text in an endangered language commonly contain translations of the text in a high-resource language. These translations typically have a practical purpose: for example, people learning an endangered language can use translations in a language they already read and speak to better understand educational and cultural material. Additionally, documentary linguists often collect translations while creating a record of the language (e.g., bilingual dictionaries, interlinear glosses, translations of speech transcriptions, etc.).

While it is difficult to estimate the number of scanned documents that contain translations, multilingual texts represent the majority in the archives we examined; AILLA includes 4,383 documents with bilingual text and 1,246 documents with trilingual text, while ELAR includes ≈5,000 multilingual documents. The structure of translations in these documents is varied, from dictionaries and interlinear glosses to multilingual books containing stories and poetry.

2.2 OCR Evaluation Dataset

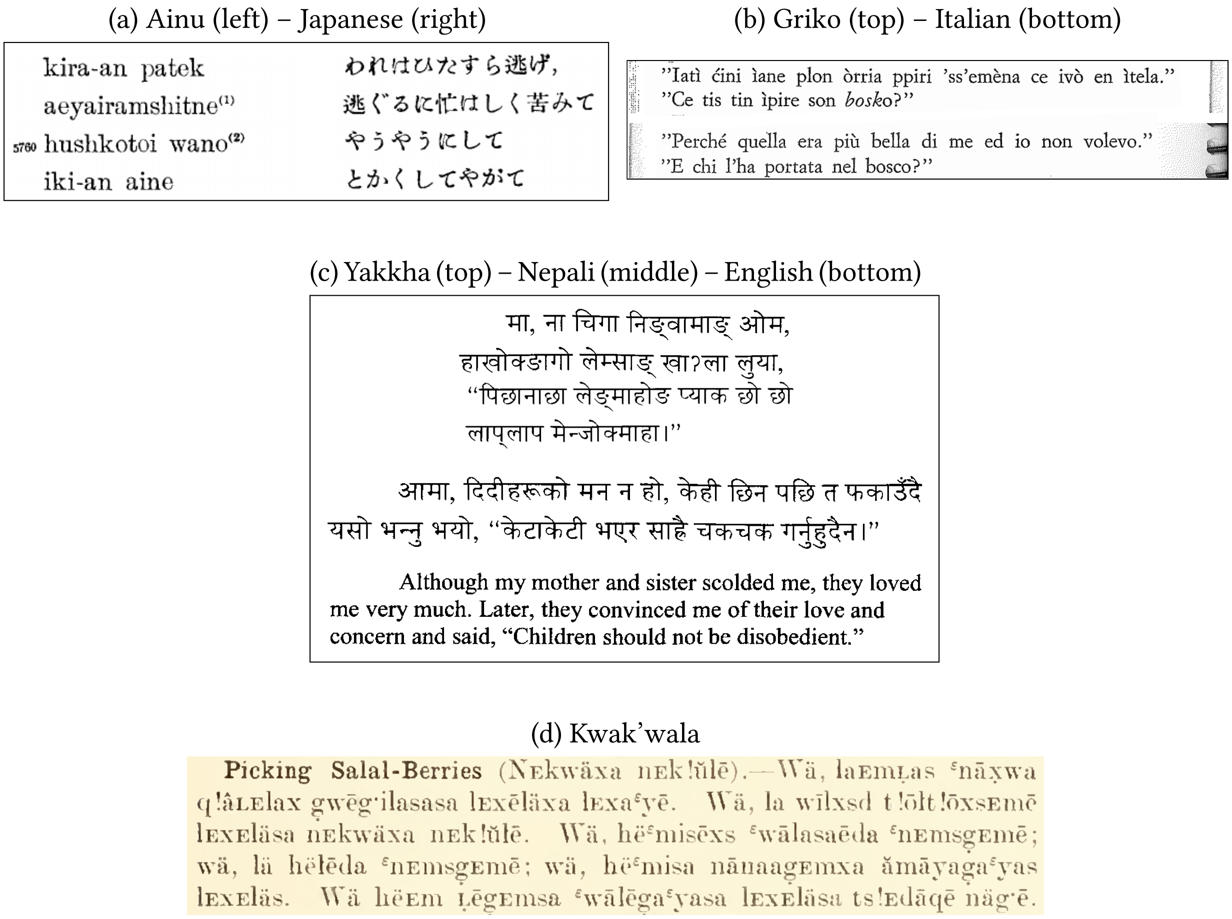
We create an OCR evaluation dataset by manually transcribing the text contained in scanned documents from four critically endangered languages[4] — Ainu, Griko, Kwak'wala, and Yakkha. These languages were chosen in an effort to create a geographically, typologically, and orthographically diverse benchmark.

The Ainu, Griko, and Yakkha documents in the dataset also contain translations of the endangered language text in various higher-resourced languages. Since we are evaluating OCR performance on the endangered language segments of the documents, we only manually transcribed the text corresponding to the endangered language content. The text corresponding to the translations is not manually transcribed. The annotated documents are described below and example images are in Figure 2.1.

[2] https://ailla.utexas.org

[3] https://elar.soas.ac.uk/

[4] UNESCO defines critically endangered languages as those where the youngest speakers are grandparents and older, and they speak the language partially and infrequently.



(a) Ainu (left) – Japanese (right)

kira-an patek	われはひたすら逃げ,
aeyairamshitne(1)	逃ぐるに忙はしく苦みて
5760 hushkotoi wano(2)	やうやうにして
iki-an aine	とかくしてやがて

(b) Griko (top) – Italian (bottom)

”Iatì ćini ìane plon òrria ppiri ’ss’emèna ce ivò en ìtela.”
”Ce tis tin ìpire son *bosk*o?”

”Perché quella era più bella di me ed io non volevo.”
”E chi l’ha portata nel bosco?”

(c) Yakkha (top) – Nepali (middle) – English (bottom)

मा, ना चिगा निङ्वामाङ् ओम,
हाखोक्ङागो लेम्साङ् खा?ला लुया,
“पिछानाछा लेङ्माहोङ प्याक छो छो
लाप्लाप मेन्जोक्माहा।”

आमा, दिदीहरूको मन न हो, केही छिन पछि त फकाउँदै यसो भन्नु भयो, “केटाकेटी भएर साहै चकचक गर्नुहुदैन।”

Although my mother and sister scolded me, they loved me very much. Later, they convinced me of their love and concern and said, “Children should not be disobedient.”

(d) Kwak’wala

Picking Salal-Berries (NEkwäxa nEk!ŭlē).—Wä, laEmLas ᵋnāx̣wa q!âLElax gwēg·ilasasa lExēläxa lExaᵋyē. Wä, la wīlxsd t!ōlt!ōxsEmē lExEläsa nEkwäxa nEk!ŭlē. Wä, hēᵋmisēxs ᵋwālasaēda ᵋnEmsgEmē; wä, lä hēlēda ᵋnEmsgEmē; wä, hēᵋmisa nānaagEmxa ămāyagaᵋyas lExEläs. Wä hëEm ḷēgEmsa ᵋwālēgaᵋyasa lExEläsa ts!Edāqē näg·ē.

Figure 2.1: Examples of scanned documents from our dataset. The endangered language text is accompanied by translations in a high-resource language in the same document for the Ainu, Griko, and Yakkha texts.

- **Ainu** is a severely endangered indigenous language from northern Japan, typically considered a language isolate. In our dataset, we use a book of Ainu epic poetry (*yukara*), with the “Kutune Shirka" yukar (Kindaichi, 1931) in Ainu transcribed in Latin script.[5] Each page in the book has a two-column structure — the left column has the Ainu text, and the right has its Japanese translation that is aligned at the line-level (see Figure 2.1 (a)). The book has 338 pages in total. Given the effort involved in annotation, we transcribe the Ainu text from 33 pages, totaling 816 lines.
- **Griko** is an endangered Greek dialect spoken in southern Italy. The language uses a combination of the Latin alphabet and the Greek alphabet as its writing system. The

[5] Some transcriptions of Ainu also use the Katakana script. See Howell (1951) for a discussion on Ainu folklore.

document we use is a book of Griko folk tales compiled by Stomeo (1980). The book is structured such that in each fold of two pages, the left page has Griko text, and the right page has the corresponding translation in Italian. Of the 175 pages in the book, we annotate the Griko text from 33 pages, resulting in 807 annotated Griko sentences.

- **Yakkha** is an endangered Sino-Tibetan language spoken in Nepal. It uses the Devanagari writing system. We use scanned images of three children's books, each of which has a story written in Yakkha along with its translation into two languages – Nepali and English (Schackow, 2012). We manually transcribe the Yakkha text from all three books. In total, we have 159 annotated Yakkha sentences.
- **Kwak'wala** is spoken on Northern Vancouver Island, nearby small islands, and the opposing mainland. The language is severely endangered, with estimates of ≈150 first-language speakers, all over the age of 70. The Kwak'wala language includes 42 consonantal phonemes (twice as many as English) and a wide range of allophonic vowels. Several writing systems exist and community preference varies between two orthographies: the U'mista and Liq'wala systems.

  However, much of the written documentation for Kwak'wala is in another orthography that was developed by anthropologist Franz Boas. The Boas orthography (Boas, 1900) was used in the extensive documentation of the Kwak'wala language and its speakers produced by Boas in collaboration with native speaker George Hunt. The Boas writing system uses Latin script characters as well as diacritics and digraphs to represent phonemic differences. Although the Boas orthography is not widely used today, the cultural and linguistic materials previously written by Hunt and Boas are of tremendous value to community-based researchers. However, they are minimally accessible since they currently exist only as non-searchable scanned images.

  In consultation with members of language revitalization projects in three Kwakiutl communities (Tsulquate, Fort Rupert, Quatsino), we focus on digitizing these significant cultural resources. We create a dataset with pages from the "Ethnology of the Kwakiutl" (Boas, 1921), containing 262 gold-transcribed lines and 2,255 unannotated lines.

2.3 Evaluation Metrics

We use two metrics for evaluating digitization performance: character error rate (CER) and word error rate (WER). Both metrics are based on edit distance and are standard for evaluating

OCR systems (Berg-Kirkpatrick et al., 2013; Schulz and Kuhn, 2017). CER is the edit distance between the predicted and the gold transcriptions of the document, divided by the total number of characters in the gold transcription. WER is similar but is calculated at the word level.

2.4 Existing OCR Methods: Promises and Pitfalls

In this section, we analyze the performance of two existing OCR methods on the endangered language documents in our evaluation dataset:

- **Google Vision**: As briefly alluded to Chapter 1, training a supervised OCR model from scratch for each endangered language is challenging, given the very small number of transcribed images we have in the dataset. Instead, we use a large-scale off-the-shelf OCR system, the underlying model of which is based on state-of-the-art supervised neural methods, to obtain a transcription on our dataset.

  The Google Vision OCR (Fujii et al., 2017; Ingle et al., 2019) is a highly-performant system that is trained on 60 languages in 27 scripts. The supported languages are primarily higher-resourced and do not include any of our target endangered languages. However, the system also provides off-the-shelf script-specific OCR models in addition to language-specific ones. Per-script models are more robust to unknown languages because they are trained on data from multiple languages and can act as a general character recognizer without relying on a single language's model. Since many endangered languages adopt standard scripts (often from the region's dominant language) as their writing systems, the per-script recognition models can potentially produce a reasonable OCR transcription for the languages in our dataset.

- **Ocular**: Unsupervised OCR methods are relatively easier to train (from scratch) for endangered languages because they do not require thousands of transcribed images for learning. We analyze the performance of Ocular (Berg-Kirkpatrick et al., 2013), an unsupervised OCR system that uses a generative model to transcribe scanned documents. Ocular's transcription model generates the image by learning the font used in the document. Ocular relies on a character n-gram language model trained on the target language. For training the language model, we use the small amount of gold-transcribed text in our dataset. Since we are evaluating the OCR performance, we use the text in a 10-fold cross-validation setup: we split the data into ten segments, using nine to train the Ocular language model and the remaining segment as the test set for OCR. The font model

	% Character Error Rate				% Word Error Rate			
OCR System	ain	grk	ybh	kwk	ain	grk	ybh	kwk
Ocular	10.49	4.58	75.60	**7.90**	47.47	15.71	99.37	**38.22**
Google Vision	**1.34**	**3.27**	**8.90**	21.12	**6.27**	**15.63**	**31.64**	82.08

Table 2.1: First pass OCR system performance. If the language's script is not covered by Google Vision (as for Kwak'wala), then Ocular results in better recognition. Otherwise, Google Vision OCR is usually significantly better.

has parameters to learn the shape of each character in the language model vocabulary. After initialization, the font parameters are updated in an unsupervised manner with the expectation-maximization algorithm, until convergence.

2.4.1 OCR Performance

Results are presented in Table 2.1. We note that although the Google OCR system is not trained on our target languages, it is trained on large amounts of data in high-resource languages that share writing systems with Ainu, Griko, and Yakkha (Latin, Greek, Devanagari scripts) and thus can recognize characters in these scripts with reasonable accuracy. We note the particularly low CER for the Ainu data, reflecting previous work that has evaluated the Google Vision system to have strong performance on typed Latin script documents (Fujii et al., 2017). On the other hand, the performance is much worse on Kwak'wala since the system has not been trained on the Boas orthography, which is unique to the Kwak'wala language. The Boas orthography uses several Latin script characters, which the system can recognize, but it also includes diacritics and digraphs unique to the writing system (a segment from the dataset is included in Figure 2.1(d)) that are incorrectly transcribed by the OCR model.

We find that the performance on Kwak'wala is considerably better with the Ocular system because the language model is trained on Kwak'wala text. Thus, unlike Google Vision, the model vocabulary contains the Boas writing system's alphabet. On the other hand, Ocular's performance on Ainu and Griko is worse than that of Google Vision, likely due to the small amount of data available for training the LM. Moreover, the performance is correlated with the *word overlap* between test data and the data used for training the language model, demonstrating Ocular's reliance on a strong language model – the word overlap is 73% for Griko, 56% for

Kwak'wala, and 48% for Ainu. Compared to higher-resource languages, the amount of text data available for training a robust language model is much smaller in the endangered language setting (i.e., millions of characters are available for languages like English and Spanish, but only a few thousand for most endangered languages).[6]

Finally, we find that Ocular does not perform well on the Yakkha dataset. This is because the design of Ocular's font model does not work with how the Devanagari script is represented in Unicode. More specifically, when a vowel diacritic is applied to a consonant in the script, the characters are combined: e.g., क + ◌ा = का. In Unicode, this is represented by two characters "क" and "◌ा", where the dotted circle is the *character combination marker* in the Unicode Standard.[7] However, since Ocular's font model operates at the character-level, it tries to generate the images of these two characters separately. Generating the diacritic "◌ा" on its own is not meaningful: the dotted circle never appears in the input image because it is supposed to be combined. Thus, the font model is unable to converge as it cannot handle character combinations when generating the image.

Even with the best-performing OCR system for each language, there remains considerable room for improvement in both CER and WER for the endangered language documents in our dataset, as compared to accuracy on higher-resourced languages Fujii et al. (2017).

Types of Errors

To better understand the challenges posed by the endangered language setting, we look that the recognition errors made by the Google Vision system on the Ainu, Griko, and Yakkha datasets. First, we look at the edit distance between the predicted and the gold transcriptions, in terms of insertion, deletion, and replacement operations. Replacement accounts for over 84% of the errors in the Griko and Ainu datasets, and 55% overall. This pattern is expected in the OCR task, as the recognition model uses the image to make predictions and is more likely to confuse a character's shape for another than to hallucinate or erase pixels. However, we observe that the errors in the Yakkha dataset do not follow this pattern. Instead, 87% of the errors for Yakkha occur because of deleted characters.

Next, we manually inspect all the errors made by the OCR system. While some errors are commonly seen in the OCR task, such as misidentified punctuation or incorrect word bound-

[6] Although Ocular has been used for lower-resourced languages like Nahuatl, previous work (Garrette et al., 2015) find that augmenting small publicly available corpora with an additional private set of transcribed documents was necessary for obtaining performance comparable to English and Spanish.

[7] https://www.unicode.org/versions/Unicode13.0.0/ch02.pdf

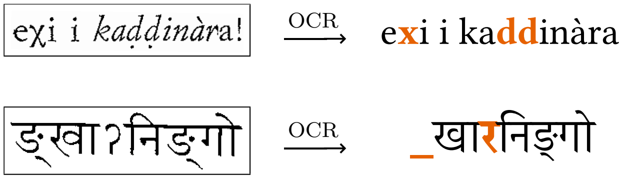


Figure 2.2: Errors in Griko (top) and Yakkha (bottom) when using the Google Vision OCR. The system frequently makes errors when scripts are mixed or when uncommon diacritics are used.

aries, 85% of the total errors occur due to specific characteristics of endangered languages that general-purpose OCR systems (such as the Google Vision tool) do not account for. Broadly, they can be categorized into two types, examples of which are shown in Figure 2.2:

- **Mixed scripts** The existing scripts that most endangered languages adopt as writing systems are often not ideal for comprehensively representing the language. For example, the Devanagari script does not have a grapheme for the glottal stop — as a solution, printed texts in the Yakkha language use the IPA symbol ‘ʔ’ (Schackow, 2015). Similarly, both Greek and Latin characters are used to write Griko. The Google Vision OCR is trained to detect scripts at the line-level and is not equipped to handle multiple scripts within a single word. As seen in Figure 2.2, the system does not recognize the Greek character χ in Griko and the IPA symbol ʔ in Yakkha. Mixed scripts cause 11% of the OCR errors.
- **Uncommon characters and diacritics** Endangered languages often use graphemes and diacritics that are part of the standard script but are not commonly seen in high-resource languages. Since these are likely rare in the OCR system's training data, they are frequently misidentified, accounting for 74% of the errors. In Figure 2.2, we see that the OCR system substitutes the uncommon diacritic ḍ in Griko. The system also deletes the Yakkha character ङ्, a diacriticized character that is infrequent in several other Devanagari script languages (such as Hindi).

2.5 Summary

In this chapter, we present the creation of an evaluation dataset for the task of OCR on endangered languages. The dataset contains documents in four languages – Ainu, Griko, Kwak'wala, and Yakkha. We extensively analyze the performance of existing OCR methods on the dataset with Google Vision, a large-scale off-the-shelf OCR tool, and Ocular, an unsupervised OCR method. We identify the types of errors made by these models and demonstrate that they are not robust to languages that use mixed orthographies and uncommon diacritics as well as languages for which training data is scarce. Our analysis highlights the need for improving text recognition techniques to transcribe documents with higher accuracy than existing systems even in very low-resource settings.

Chapter 3

OCR Post-Correction for Endangered Language Texts

In the previous chapter, we presented the creation of a benchmark dataset of transcriptions for scanned books in four critically endangered languages and conducted a systematic analysis of how existing OCR methods are not robust to the data-scarce setting of endangered languages.

To improve performance on endangered languages, in this chapter, we focus on developing a method for better OCR performance in extremely low-resource scenarios. More specifically, we present a model for *OCR post-correction* — instead of training an OCR system from scratch, which requires significant manual transcription or large text corpora to obtain high-quality transcriptions, the goal of post-correction is to fix recognition errors in already existing OCR outputs, such as those from the two systems presented in the previous chapter (i.e., Google Vision and Ocular). An example of post-correction is in Figure 1.1.

The task of OCR post-correction is relatively well-studied and most state-of-the-art post-correction methods use neural sequence-to-sequence models (i.e., the encoder-decoder framework). These methods typically rely on considerable resources in the target language, such as a large number of pairs of erroneous and manual corrected transcriptions for supervised learning (Schnober et al., 2016; Rigaud et al., 2019) or substantial textual data to train a language model that conditions unsupervised post-correction (Krishna et al., 2018; Dong and Smith, 2018). While these resources are readily available for high-resource languages, these resources are severely limited in endangered languages, preventing the direct application of existing post-correction methods in our setting.

In this chapter, we present a post-correction method designed for under-resourced languages. Our method, compounding on previous neural models for the task, makes three improvements tailored to learning a strong model even when very minimal training data is available. First, we use a **multi-source model** to incorporate information from the high-resource translations that commonly appear in endangered language books, as seen in Section 2.2. Next, we introduce **structural biases** within the model to ease learning from small amounts of data. Finally, we add **pretraining** to utilize the little unannotated data that exists in endangered languages. In experiments on our evaluation dataset, we demonstrate that our proposed post-correction method reduces character and word error rates across the four languages. We also extensively analyze which factors within the model contribute to the improvements in performance. The method presented in this chapter has also been published in Rijhwani et al. (2020).

3.1 Problem Formulation

In this section, we formulate the tasks of OCR and OCR post-correction and introduce notation that will be used throughout the thesis.

Optical Character Recognition OCR systems are trained to find the best transcription corresponding to the text in an image. We use the two OCR systems (detailed in Section 2.4) to produce a *first pass transcription* of the endangered language text in the images in our dataset. Let the first pass transcription for a data point (e.g., a line or sentence) be a sequence of characters $\boldsymbol{x} = [x_1, \ldots, x_N]$.

OCR Post-Correction The goal of post-correction is to reduce recognition errors in the first pass transcription, which are frequent in our setting because of the lack of OCR training data in the target endangered languages. The correction model takes $\boldsymbol{x}$ as input and produces the *corrected transcription* of the endangered language document, a sequence of characters $\boldsymbol{y} = [y_1, \ldots, y_K]$.

$$\boldsymbol{y} = \underset{\boldsymbol{y}'}{\operatorname{argmax}}\, p_{\text{corr}}(\boldsymbol{y}'|\boldsymbol{x})$$

Incorporating Translations We use information from high-resource translations of the endangered language text. As seen in Section 2.2, these translations are commonly found in endangered language documents. Such translations exist for all the documents in our evaluation dataset. We do not have gold transcriptions for the translations in the documents. Instead, we use the Google Vision OCR system (Section 2.4), which is trained on many high-resource languages, to obtain a transcription of the scanned translation. Let this transcription be a sequence of characters $\boldsymbol{t} = [t_1, \ldots, t_M]$. In our method, we use $\boldsymbol{t} = [t_1, \ldots, t_M]$ as additional input to condition the post-correction model:

$$\boldsymbol{y} = \underset{\boldsymbol{y}'}{\operatorname{argmax}}\, p_{\text{corr}}(\boldsymbol{y}'|\boldsymbol{x}, \boldsymbol{t})$$

3.2 OCR Post-Correction Model

In this section, we describe our proposed OCR post-correction model. The base architecture of the model is a multi-source sequence-to-sequence framework (Zoph and Knight, 2016; Libovický and Helcl, 2017) that uses an LSTM encoder-decoder model with attention (Bahdanau et al., 2015). We propose improvements to training and modeling for the multi-source architecture, specifically tailored to ease learning in data-scarce settings.

3.2.1 Multi-source Architecture

Our post-correction formulation takes as input the first pass OCR of the endangered language segment $\boldsymbol{x}$ and the OCR of the translated segment $\boldsymbol{t}$, to predict an error-free endangered language text $\boldsymbol{y}$. The model architecture is shown in Figure 3.1.

The model consists of two encoders — one that encodes $\boldsymbol{x}$ and one that encodes $\boldsymbol{t}$. Each encoder is a character-level bidirectional LSTM (Hochreiter and Schmidhuber, 1997) and transforms the input sequence of characters into a sequence of hidden state vectors: $\mathbf{h}^x$ for the endangered language text and $\mathbf{h}^t$ for the translation.

The model uses an attention mechanism during the decoding process to use information from the encoder's hidden states. We compute the attention weights over each of the two encoders independently. At the decoding time step k:

$$e^x_{k,i} = \mathbf{v}^x \tanh\left(\mathbf{W}^x_1 \mathbf{s}_{k-1} + \mathbf{W}^x_2 \mathbf{h}^x_i\right) \tag{3.1}$$

$$\boldsymbol{\alpha}^x_k = \operatorname{softmax}\left(\mathbf{e}^x_k\right)$$

$$\mathbf{c}^x_k = \left[\Sigma_i \alpha^x_{k,i} \mathbf{h}^x_i\right]$$

where $\mathbf{s}_{k-1}$ is the decoder state of the previous time step and $\mathbf{v}^x$, $\mathbf{W}^x_1$ and $\mathbf{W}^x_2$ are trainable parameters. The encoder hidden states $\mathbf{h}^x$ are weighted by the attention distribution $\boldsymbol{\alpha}^x_k$ to produce the context vector $\mathbf{c}^x_k$. We follow a similar procedure for the second encoder to produce

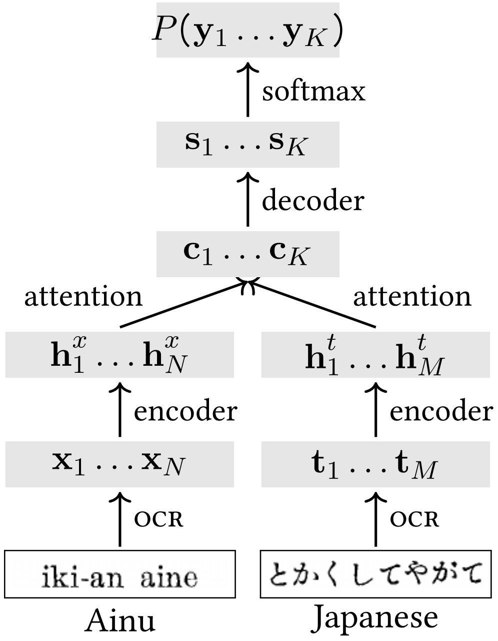

$P(\mathbf{y}_1 \ldots \mathbf{y}_K)$
softmax
$\mathbf{s}_1 \ldots \mathbf{s}_K$
decoder
$\mathbf{c}_1 \ldots \mathbf{c}_K$
attention
attention
$\mathbf{h}^x_1 \ldots \mathbf{h}^x_N$
$\mathbf{h}^t_1 \ldots \mathbf{h}^t_M$
encoder
encoder
$\mathbf{x}_1 \ldots \mathbf{x}_N$
$\mathbf{t}_1 \ldots \mathbf{t}_M$
OCR
OCR
iki-an aine
とかくしてやがて
Ainu
Japanese

Figure 3.1: The proposed multi-source architecture with the encoder for an endangered language segment (left) and an encoder for the translated segment (right). The input to the encoders is the first pass OCR over the scanned images of each segment. For example, the OCR on the scanned images of some Ainu text (left) and its Japanese translation (right).

$\mathbf{c}^t_k$. We concatenate the context vectors to combine attention from both sources (Zoph and Knight, 2016):

$$\mathbf{c}_k = \left[\mathbf{c}^x_k; \mathbf{c}^t_k\right]$$

$\mathbf{c}_k$ is used by the decoder LSTM to compute the next hidden state $\mathbf{s}_k$ and subsequently, the probability distribution for predicting the next character $\mathbf{y}_k$ of the target sequence $\boldsymbol{y}$:

$$\mathbf{s}_k = \mathrm{lstm}\left(\mathbf{s}_{k-1}, \mathbf{c}_k, \mathbf{y}_{k-1}\right) \tag{3.2}$$

$$P\left(\mathbf{y}_k\right) = \mathrm{softmax}\left(\mathbf{W}\mathbf{s}_k + \mathbf{b}\right) \tag{3.3}$$

Training and Inference The model is trained for each language with the cross-entropy loss ($\mathcal{L}_{\text{ce}}$) on the small amount of transcribed data we have. Beam search is used at inference time.

3.2.2 Model and Training Improvements

With the minimal annotated data we have, it is challenging for the neural network to learn a good distribution over the target characters. We propose a set of adaptations to the base architecture that improves the post-correction performance without additional annotation. The adaptations are based on characteristics of the OCR task itself and the errors made by the upstream OCR tool as analyzed in Section 2.4.

Diagonal attention loss As discussed in Section 2.4, substitution errors are more frequent in the OCR task than insertions or deletions; consequently, we expect the source and target to have similar lengths. Moreover, post-correction is a monotonic sequence-to-sequence task, and reordering rarely occurs (Schnober et al., 2016). Hence, we expect attention weights to be higher at characters close to the diagonal for the endangered language encoder.

We modify the model such that all the elements in the attention vector that are not within j steps (we use $j = 3$) of the current time step k are added to the training loss, thereby encouraging elements away from the diagonal to have lower values. The diagonal loss summed over all time steps for a training instance, where N is the length of $\boldsymbol{x}$, is:

$$\mathcal{L}_{\text{diag}} = \sum_{k} \left(\sum_{i=1}^{k-j} \alpha_{k,i}^{x} + \sum_{i=k+j}^{N} \alpha_{k,i}^{x} \right)$$

Copy mechanism The performance of the first pass OCR systems Table 2.1 indicates that the first pass OCR predicts a majority of the characters accurately. In this scenario, enabling the model to directly copy characters from the first pass OCR rather than generate a character at each time step might lead to better performance (Gu et al., 2016). We incorporate the copy mechanism proposed in See et al. (2017). The mechanism computes a “generation probability” at each time step k, which is used to choose between *generating* a character (Equation 3.3) or *copying* a character from the source text by sampling from the attention distribution $\boldsymbol{\alpha}_{k}^{x}$.

Coverage Given the monotonicity of the post-correction task, the model should not attend to the same character repeatedly. However, repetition is a common problem for neural encoder-decoder models (Mi et al., 2016; Tu et al., 2016). To account for this problem, we adapt the coverage mechanism from See et al. (2017), which keeps track of the attention distribution over past time steps in a coverage vector. For time step k, the coverage vector would be $\mathbf{g}_k = \sum_{k'=0}^{k-1} \boldsymbol{\alpha}_{k'}^{x}$. $\mathbf{g}_k$ is used as an extra input to the attention mechanism, ensuring that future attention decisions take the weights from previous time steps into account. Equation 3.1, with learnable parameter $\mathbf{w}_g$, becomes:

$$e_{k,i}^{x} = \mathbf{v}^{x} \tanh\left(\mathbf{W}_1^{x}\mathbf{s}_{k-1} + \mathbf{W}_2^{x}\mathbf{h}_i^{x} + \mathbf{w}_g g_{k,i}\right)$$

g_k is also used to penalize attending to the same locations repeatedly with a coverage loss. The coverage loss summed over all time steps k is:

$$\mathcal{L}_{\text{cov}} = \sum_{k} \sum_{i=1}^{n} \min\left(\alpha_{k,i}^{x}, g_{k,i}\right)$$

Therefore, with our model adaptations, the loss for a single training instance including the cross-entropy loss, the diagonal attention loss, and the coverage loss is:

$$\mathcal{L} = \mathcal{L}_{\text{ce}} + \mathcal{L}_{\text{diag}} + \mathcal{L}_{\text{cov}} \tag{3.4}$$

3.2.3 Utilizing Untranscribed Data

As discussed in Section 2.2, given the effort involved, we transcribe only a subset of the pages in each scanned book. Nonetheless, we leverage the remaining unannotated pages for pretraining our model. We use the upstream OCR tool to get a first pass transcription on all the unannotated pages. We then create "pseudo-target" transcriptions for the endangered language text as described below:

- **Denoising rules** Using a small fraction of the available annotated pages, we compute the edit distance operations between the first pass OCR and the gold transcription. The operations of each type (insertion, deletion, and replacement) are counted for each character and divided by the number of times that character appears in the first pass OCR. This forms a probability of how often the operation should be applied to that specific character.

  We use these probabilities to form rules, such as p(replace d with ḍ)=0.4 for Griko and p(replace ? with ʔ)=0.7 for Yakkha. These rules are applied to remove errors from, or "denoise", the first pass OCR transcription.

- **Sentence alignment** We use Yet Another Sentence Aligner (Lamraoui and Langlais, 2013) for unsupervised alignment of the endangered language and translation on the unannotated pages.

Given the aligned first pass OCR for the endangered language text and the translation along with the pseudo-target text, $\boldsymbol{x}$, $\boldsymbol{t}$ and $\hat{\boldsymbol{y}}$ respectively, the pretraining steps, in order, are:

- **Pretraining the encoders** We pretrain both the forward and backward LSTMs of each encoder with a character-level language model objective: the endangered language encoder on $\boldsymbol{x}$ and the translation encoder on $\boldsymbol{t}$.

- **Pretraining the decoder** The decoder is pretrained on the pseudo-target $\hat{\boldsymbol{y}}$ with a character-level language model objective.
- **Pretraining the seq-to-seq model** The model is pretrained with $\boldsymbol{x}$ and $\boldsymbol{t}$ as the sources and the pseudo-target $\hat{\boldsymbol{y}}$ as the target transcription, using the post-correction loss function $\mathcal{L}$ as defined in Equation 3.4.

3.3 Experiments

This section discusses our experimental setup and the post-correction performance on the four endangered languages on our dataset.

3.3.1 Experimental Setup

Data Splits We use the benchmark dataset described in Section 2.2 for evaluating the performance of our method. We perform 10-fold cross-validation for all experimental settings because of the small size of the datasets. For each language, we divide the transcribed data into 11 segments — we use one segment for creating the *denoising rules* described in the previous section and the remaining ten as the folds for cross-validation. In each cross-validation fold, eight segments are used for training, one for validation, and one for testing.

We divide the dataset at the page-level for the Ainu, Griko, and Kwak'wala documents. This results in 11 segments with three pages each for Ainu and Griko and 11 segments with a single page each for Kwak'wala. For the Yakkha documents, we divide at the paragraph-level, due to the small size of the dataset. We have 33 paragraphs across the three books in our dataset, resulting in 11 segments that contain three paragraphs each. The multi-source results for Yakkha reported in Table 3.1 use the English translations. Results with Nepali are similar and are excluded for clarity.

Metrics We use two metrics for evaluating the digitization performance of all compared systems: character error rate (CER) and word error rate (WER), as described in Section 2.3.

Methods In our experiments, we compare the performance of our proposed method with the first pass OCR and with two systems from recent work in OCR post-correction. All the post-correction methods have two variants – the single-source model with only the endangered language encoder and the multi-source model that additionally incorporates translations from the documents using the high-resource translation encoder.

- Fp-Ocr: The first pass transcription obtained from Ocular for Kwak'wala and from the Google Vision OCR system for Ainu, Griko, and Yakkha (see Section 2.4 for details about the performance of each OCR system).
- Base: This system is the base sequence-to-sequence architecture described in Section 3.2.1. The single-source variant of this model has been used in recent work on OCR post-correction, resulting in state-of-the-art performance on high-resource languages (Hämäläinen and Hengchen, 2019). Both the single-source and multi-source variants of this system are used for English OCR post-correction in Dong and Smith (2018).
- Copy: This system is the base architecture with a copy mechanism as described in Section 3.2.2. The single-source variant of this model is used for OCR post-correction on Romanized Sanskrit in Krishna et al. (2018).[1]
- Ours: The model with all the adaptations proposed in Section 3.2.2 and Section 3.2.3.

Implementation The post-correction models are implemented using the DyNet neural network toolkit (Neubig et al., 2017), and all reported results are the average of five training runs with different random seeds. We assume knowledge of the entire alphabet of the endangered language for all the methods, which is straightforward to obtain for most languages. The decoder's vocabulary contains all these characters, irrespective of their presence in the training data, with corresponding randomly-initialized character embeddings.

The hyperparameters used are:

- Character embedding size = 128
- Number of LSTM layers = 1
- Hidden state size of the LSTM = 256
- Attention size = 256
- Beam size = 4
- For the diagonal loss, $j = 3$
- Minibatch size for training = 1
- Maximum number of epochs = 150

[1] Although Krishna et al. (2018) use BPE tokenization, experiments showed that character-level models result in much better performance on our dataset, likely due to the limited data available for training the BPE model.

	Character Error Rate							
	Ainu		Griko		Yakkha		Kwak'wala	
Model	Multi	Single	Multi	Single	Multi	Single	Multi	Single
Fp-Ocr	–	1.34	–	3.27	–	8.90	–	7.90
Base	1.56	1.41	6.78	5.95	70.39	71.71	–	70.62
Copy	2.04	1.99	2.54	2.28	14.77	12.30	–	14.84
Ours	0.92	**0.80**	**1.66**	1.70	**7.75**	8.44	–	**4.97**

	Word Error Rate							
	Ainu		Griko		Yakkha		Kwak'wala	
Model	Multi	Single	Multi	Single	Multi	Single	Multi	Single
Fp-Ocr	–	6.27	–	15.63	–	31.64	–	38.22
Base	8.56	7.88	15.13	13.67	98.15	99.10	–	89.57
Copy	9.48	8.57	9.33	8.90	30.36	27.81	–	48.81
Ours	5.75	**5.19**	**7.46**	7.51	**20.95**	21.33	–	**27.65**

Table 3.1: Our method improves performance over all baselines (10-fold cross-validation averaged over five randomly seeded runs). We present multi- and single-source variants and **highlight** the best model for each language.

- Patience for early stopping = 10 epochs
- Pretraining epochs for encoder/decoder = 10
- Pretraining epochs for seq-to-seq model = 5

We use the same values of the hyperparameters for each language and all the systems. We select the best model with early stopping on the character error rate of the validation set.

3.3.2 Main Results

Table 3.1 shows the performance of the baselines and our proposed method for each language. Overall, our method results in an improved CER and WER over existing methods across all four languages in the evaluation dataset.

The Base system does not improve the recognition rate over the first pass transcription,

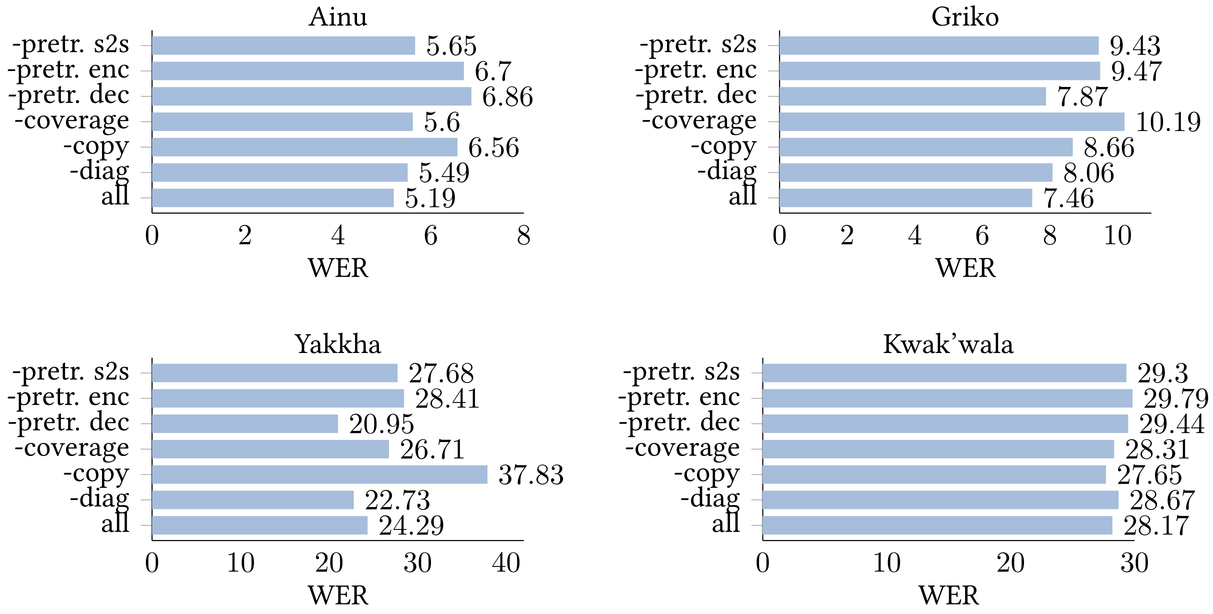


Figure 3.2: WER with model component ablations on the best model setting in Table 3.1. “all" includes all the adaptations we propose. Each ablation removes a single component from the “all" model, e.g. “-pretr. s2s" removes the seq-to-seq model pretraining.

apart from a small decrease in the Griko WER. The performance on Yakkha and Kwak'wala, particularly, is significantly worse than Fp-Ocr: likely because the training data size is much smaller in these languages than that of Griko and Ainu, and the model is unable to learn a reasonable distribution. However, on adding a copy mechanism to the base model in the Copy system, the performance is notably better than Base for Griko, Kwak'wala, and Yakkha. This indicates that adaptations to the base model that cater to specific characteristics of the post-correction task can alleviate some of the challenges of learning from small amounts of data.

Both the single-source and the multi-source variants of our proposed method (Ours) improve over the baselines, demonstrating that the model adaptations can increase recognition accuracy even without high-resource translations. We see that using the high-resource translations results in better post-correction performance for Griko and Yakkha, but the single-source model achieves better accuracy for Ainu. We attribute this to two factors: the very low error rate of the first pass transcription for Ainu and the relatively high error rate (based on manual inspection) of the OCR on the Japanese translation. Despite being a high-resource language, OCR is difficult due to the complexity of Japanese characters and low scan quality. The noise resulting from Japanese OCR errors likely affects performance in the multi-source setup.

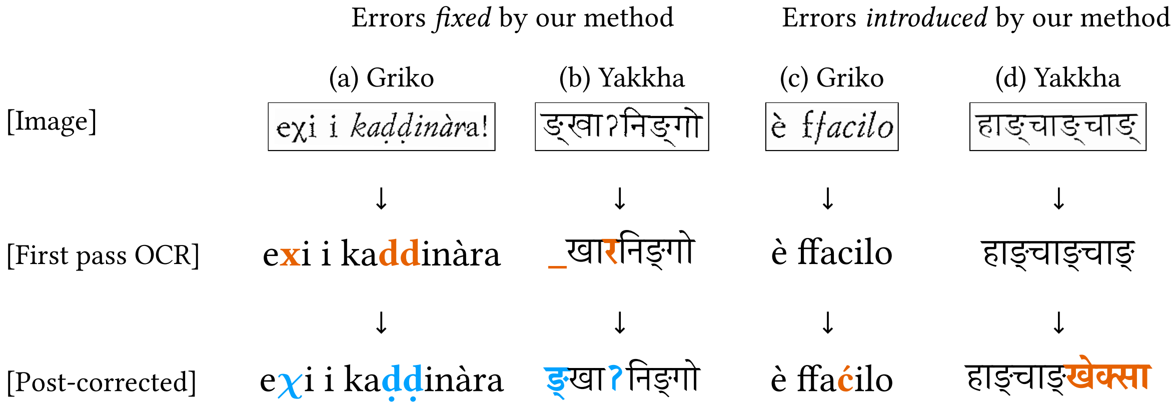


Figure 3.3: Our model fixes many mixed script and uncommon diacritics errors such as (a) and (b). In rare cases, it "over-corrects" the first pass OCR transcription, introducing errors such as (c) and (d).

3.3.3 Ablation Studies

Next, we study the effect of our proposed adaptations and evaluate their benefit to the performance of each language. Figure 3.2 shows the word error rate with models that remove one adaptation from the model with all the adaptations ("all").

For Ainu and Griko, removing any single component increases the WER, with the complete ("all") method performing the best. There is little variance in performance with ablations on the Kwak'wala language dataset. Our proposed adaptations add the most benefit for Yakkha, which has the fewest training data and relatively less accurate first pass OCR. The copy mechanism is crucial for good performance, but removing the decoder pretraining ("pretr. dec") leads to the best scores among all the ablations. The denoising rules used to create the pseudo-target data for Yakkha are likely not accurate since they are derived from only three paragraphs of annotated data. Consequently, using it to pretrain the decoder leads to a poor language model.

3.3.4 Error Analysis

We systematically inspect all the recognition errors in the output of our post-correction model to determine the sources of improvement with respect to the first pass OCR. We also examine the types of errors introduced by the post-correction process.

We observe a *91% reduction* in the number of errors due to mixed scripts and a *58% reduction* in the errors due to uncommon characters and diacritics (as defined in Section 2.4). Examples of these are shown in Figure 3.3 (a) and (b): mixed script errors such as the χ character in Griko and the glottal stop ʔ in Yakkha are successfully corrected by the model. The model is also able to correct uncommon character errors like **ḍ** in Griko and **ङ्** in Yakkha.

Examples of errors introduced by the model are shown in Figure 3.3 (c) and (d). Example (c) is in Griko, where the model incorrectly adds a diacritic to a character. We attribute this to the fact that the first pass OCR does not recognize diacritics well; hence, the model learns to add diacritics frequently while generating the output. Example (d) is in Yakkha. The model inserts several incorrect characters, and can likely be attributed to the lack of a good language model due to the relatively smaller amount of training data we have in Yakkha.

3.4 Related Work

Post-correction for OCR is well-studied for high-resource languages. Early approaches include lexical methods and weighted finite-state methods (see Schulz and Kuhn (2017) for an overview). Recent work has primarily focused on using neural sequence-to-sequence models. Hämäläinen and Hengchen (2019) use a BiLSTM encoder-decoder with attention for historical English post-correction. Similar to our base model, Dong and Smith (2018) uses a multi-source model to combine the first pass OCR from duplicate documents in English. There has been little work on lower-resourced languages. Kolak and Resnik (2005) present a probabilistic edit distance-based post-correction model applied to Cebuano and Igbo, and Krishna et al. (2018) show improvements on Romanized Sanskrit OCR by adding a copy mechanism to a neural sequence-to-sequence model.

Multi-source encoder-decoder models have been used for various tasks including machine translation (Zoph and Knight, 2016; Libovický and Helcl, 2017) and morphological inflection (Kann et al., 2017; Anastasopoulos and Neubig, 2019). Perhaps most relevant to our work is the multi-source model presented by Anastasopoulos and Chiang (2018), which uses high-resource translations to improve speech transcription of lower-resourced languages.

Finally, Bustamante et al. (2020) construct corpora for four endangered languages from text-based PDFs using rule-based heuristics. Data creation from such unstructured text files is an important research direction, complementing our post-correction method for improving text extraction accuracy for scanned documents.

3.5 Summary

In this chapter, we develop a model for OCR post-correction that, unlike previous work, requires a very small amount of manually annotated data for training. The method uses character-level LSTMs in a sequence-to-sequence architecture, with several adaptations to improve performance in low-resource settings: a multi-source encoder to include high-resource translations of the text, structural biases in the model, and pretraining to use unlabeled data. The proposed method reduces the character error rate by 34.6% and the word error rate by 32.4% averaged over the four endangered languages in our benchmark dataset. We also conduct a systematic ablation study, demonstrating the utility of our model adaptations. Additionally, we present a comprehensive error analysis that showed our model fixes a large majority of the errors caused by specific characteristics of endangered languages (mixed scripts and uncommon diacritics).

Chapter 4

Efficient Use of Unlabeled Documents

In Chapter 3, we demonstrated that *post-correction* improves the performance of existing OCR systems on endangered languages, where we adapt neural encoder-decoder models for post-correction in the less-well-resourced endangered languages setting by adding translations and structural biases to the model. However, even with such methods targeted at low-resource learning, post-correction performance is still dependent on manually curated data, which are minimally available for most endangered languages. On the other hand, unannotated raw images that need to be digitized are relatively less scarce; for many endangered languages, hundreds of printed pages exist, with only a small subset manually transcribed. In this chapter, we present a semi-supervised learning method for OCR post-correction that efficiently utilizes these unannotated pages to improve performance.

The method has two key components. We first present a **self-training** technique for OCR post-correction to create *pseudo*-training data. A baseline post-correction model is used to correct the initial OCR output on the unannotated pages, and the generated "post-corrected" text is then used as *pseudo*-training data to improve the model. This process is repeated to iteratively obtain better predictions on the unannotated pages.

While self-training is a straightforward way to use the unannotated data, incorrect predictions in the *pseudo*-training data may introduce noise into the model (Zhu and Goldberg, 2009). To counterbalance the influence of this noise and enforce consistency in the recognized vocabulary, we propose **lexically-aware decoding**, an inference strategy that encourages the model to generate predictions that contain "known" words. We use the *pseudo*-training data to train a count-based language model, represented with a weighted finite-state automaton (WFSA) for efficient and effective decoding. Our decoding method jointly uses an LSTM decoder and the WFSA to make OCR post-correction predictions.

The intuition behind the joint decoding strategy is simple. As the model iteratively improves with self-training, the quality of the *pseudo*-training data is also likely to improve and contain an increasing number of correctly predicted words, resulting in a better count-based language model. Consequently, joint decoding reinforces the prediction of more accurate words and mitigates the noise introduced by incorrect words in the *pseudo*-training data.

The semi-supervised method reduces the character and word error rates by 15%–29% over the OCR post-correction method for endangered languages presented in Chapter 3. We find that the combination of self-training and lexically-aware decoding is essential for achieving consistent improvements in performance across all languages. The method presented in this chapter has also been published in Rijhwani et al. (2021).

4.1 Base Model

We use the OCR post-correction formulation as described in Section 3.1. For the semi-supervised learning method, we require a baseline post-correction model to obtain initial predictions on the unlabeled data.

For the baseline model, we use the technique described in the previous chapter: a sequence-to-sequence model that uses an attention-based LSTM encoder-decoder (Bahdanau et al., 2015), with adaptations for low-resource OCR post-correction.[1]

Recall that the model is trained in a supervised manner with a small number of manual transcriptions: the training data includes pairs of first pass OCR text with its corresponding error-free transcription. The post-correction training loss function (denoted as $\mathcal{L}$; defined in Equation 3.4) is a combination of cross-entropy loss along with the diagonal attention loss and the coverage loss from the structural biases. Similar to many sequence generation methods, inference with a trained model is performed using beam search.

In the following sections, we use this model as the base model for our proposed semi-supervised learning technique for OCR post-correction. Given the minimal manually transcribed data we have in endangered languages, the method aims to efficiently use the relatively larger number of pages without gold transcriptions to improve performance. To this end, we introduce two methodological improvements: (1) self-training and (2) lexically-aware decoding.

[1] The model in Chapter 3 incorporate translations into the model with a multi-source encoder. We omit this from the semi-supervised formulation, considering applicability to texts without available translations. However, adding an encoder into the framework remains straightforward and can be used if translations exist.

4.2 Self-Training

Self-training is a semi-supervised learning method, where a trained model is used to make predictions on unlabeled data, and the model is then retrained on its own predictions (Zhu and Goldberg, 2009).[2]

Consider that we have a set of images with manually created transcriptions and a set of images without gold transcriptions. We can obtain a first-pass transcription for the text contained in the images (both sets) with existing OCR tools.

More formally, we have a gold-transcribed dataset $D = \{\langle \boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)} \rangle\}_{i=1}^{d}$, where $\boldsymbol{x}^{(i)}$ is the first pass transcription and $\boldsymbol{y}^{(i)}$ is the error-free manual transcription of the ith training instance.[3] We also have a dataset for which only the first pass OCR is available (i.e., no manual transcriptions), $U = \{\boldsymbol{x}^{(j)}\}_{j=1}^{u}$. For most cases in the endangered languages setting, the set without gold transcriptions is much larger, that is, $u \gg d$.

Since self-training requires a baseline model to get an initial set of predictions on U, we first train the base model described in Section 3.2.1. Let the trained base model be f_θ. Next, we use the predictions on U from f_θ to self-train the model. We follow the self-training strategy recommended in He et al. (2020), which involves two steps: "pseudo-training" and "fine-tuning". We describe each step of the self-training procedure in detail below:

1. Apply a trained OCR post-correction model f_θ on each instance in the set U to obtain predictions using beam search inference.

   For an instance $\boldsymbol{x}$, let the prediction be $f_\theta(\boldsymbol{x})$.

2. Create a *pseudo-annotated dataset* with the predictions from step 1. Let this be $S = \{\langle \boldsymbol{x}, f_\theta(\boldsymbol{x}) \rangle \mid \boldsymbol{x} \in U\}$.

3. Pseudo-train the model f_θ on sets U and S.

   In this step, we first pseudo-train the encoder and the decoder components, and then pseudo-

[2] While this method is typically called self-training, it is also sometimes called "hard-EM" (Spitkovsky et al., 2010). Using the traditional (*soft*) EM algorithm is computationally infeasible with our model because calculating the expectation requires generating all possible sequence predictions on the unannotated pages in the dataset. Although sampling can be used to approximate EM in this case, because we cannot use dynamic programming with our model, we would still need to sample a large number of outputs to get robust counts for EM. Instead, we use beam search to generate a single best output ("hard-EM"), which we then use for self-training.

[3] In our dataset, the source and target data instances are either at the line-level or the sentence-level (see the description of the data in Chapter 2).

train the end-to-end post-correction model. The pseudo-training procedure is as follows:

a) Train the encoder with a character-level language modeling (LM) objective on U.

   As discussed in Section 3.2.1, the encoder component of the model is an LSTM that operates at the character-level. We pseudo-train this LSTM with a language model objective on each text sequence $\boldsymbol{x} \in U$.

   That is, at each timestep t, the LSTM is trained to predict the next character in the input sequence. Given a sequence of characters $\boldsymbol{x} = [x_1, \ldots, x_N]$, the training objective maximizes $\prod_{t=1}^{N} P(x_t \mid x_1, \ldots, x_{t-1})$.

   This is the standard LM objective function and has been proven helpful for pretraining LSTMs to improve hidden representations (Dai and Le, 2015; Ramachandran et al., 2017).

b) Train the decoder LSTM with the LM objective described above, using the baseline model's predictions $\{f_\theta(\boldsymbol{x}) \mid \boldsymbol{x} \in U\}$.

c) Train the sequence-to-sequence model on the *pseudo-annotated dataset* S with the post-correction loss function $\mathcal{L}$ from Section 3.2.1.

4. Given the pseudo-trained model f_θ, fine-tune the model on the gold-transcribed dataset D, with the loss function $\mathcal{L}$.

5. Repeat step 1 to step 4 until convergence or for the maximum iterations permitted.

As indicated above, self-training is a straightforward semi-supervised technique to leverage documents without gold transcriptions to improve OCR post-correction performance. We note that some self-training methods (Yarowsky, 1995; Lee et al., 2013; Zoph et al., 2020, *inter alia*) replace steps 4 and 5 with a single step that trains f_θ on $S \cup D$. However, this led to slightly worse performance in our preliminary experiments. We also observed that pseudo-training the LSTMs with an LM objective (steps 3(a) and 3(b) above) is necessary for good performance and that applying the self-training steps on f_θ from the previous iteration led to better results than re-initializing the model.[4]

Further, as recommended in He et al. (2020) to improve self-training for neural sequence generation, we add a *dropout* layer into the base model at the encoder and decoder hidden states during pseudo-training and fine-tuning (steps 3 and 4).

[4] In preliminary experiments, we also tried using $S \cup D$ in step 3(c). However, the post-correction performance was approximately the same as using only the set S.

4.3 Lexically-Aware Decoding

Although self-training is a simple approach that leads to improvements in post-correction performance without additional manual annotation, incorrect predictions in the pseudo-annotated data may introduce noise into the model, potentially reinforcing the errors in the next self-training iteration (Zhu and Goldberg, 2009). Such noise is more likely to occur in the endangered languages setting, where the base model is trained on minimal data and, thus, sometimes generates erroneous predictions.

While some self-training methods use confidence scores to remove noisy predictions (such as Yarowsky (1995)), these are typically designed for classification tasks. Designing such heuristics is challenging for OCR post-correction because the predictions are generated at the character-level; specific characters may be incorrect, but discarding the entire predicted sequence (i.e., the line or sentence) is inefficient, particularly in a low-resource scenario. To mitigate these issues, we propose *lexically-aware decoding*, an inference strategy based on our observations of the challenges associated with the OCR post-correction task.

More specifically, our preliminary experiments with self-training indicated that the errors made by the model are typically inconsistent. For a particular word, some instances may be correctly predicted by the model. For the instances of the word that are incorrect, we observe that they are likely to be erroneous in different ways, i.e., different subsets of characters in the word are incorrectly predicted. This is expected since the same word can appear in varied contexts, or the first pass OCR for the word can differ. Our empirical observations on the pseudo-annotated dataset S showed that, since the errors are inconsistent, the correct form of the word is more frequent than incorrect forms. *Lexically-aware decoding* is designed to influence the OCR post-correction model to generate words that frequently occur in the set S, in the expectation that these are correct word forms.

We first describe the construction of a model that accounts for word frequency in the predictions along with a character n-gram model to enable the prediction of unseen words. Then, we present a joint decoding method that uses the frequency-based models in combination with the LSTM decoder for improved OCR post-correction.

4.3.1 Count-Based Language Model

From the self-training method in Section 4.2, we have a pseudo-annotated dataset $S = \{\langle \boldsymbol{x}, f_\theta(\boldsymbol{x}) \rangle \mid \boldsymbol{x} \in U\}$, where $f_\theta(\boldsymbol{x})$ is the model's prediction for input sequence $\boldsymbol{x}$. We train a count-based

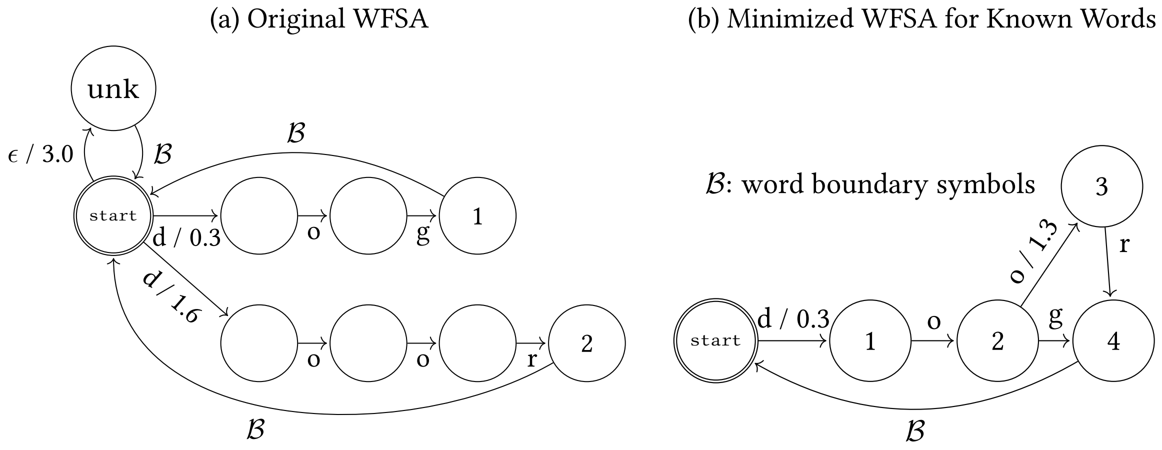


Figure 4.1: The (a) WFSA and (b) minimized WFSA we construct, for a hypothetical language model with a two word vocabulary: $P(\text{dog}) = 0.75$; $P(\text{door}) = 0.2$; $P(\text{<unk>}) = 0.05$. The transition weights are negative log probabilities. In (b), for simplicity, we show only the known word states after determinization and minimization.

word-level unigram language model (LM) on $\{f_\theta(\boldsymbol{x}) \mid \boldsymbol{x} \in U\}$. The LM is built by computing frequency-based probabilities for each word found in the predictions. However, we have to reserve some probability mass to account for unknown words (words unseen in the predictions).

We use modified Kneser-Ney smoothing to derive the unknown word (“<unk>") probability. Since we use a unigram LM, the smoothing process is similar to absolute discounting. However, we use the discount values based on the modified Kneser-Ney method, which are derived from word counts in the dataset, as opposed to using a fixed discount value (Chen and Goodman, 1999). We denote the probability from the smoothed LM for a known word w as $p_{\text{word}}(w)$ and the unknown word probability as $p_{\text{word}}(\text{<unk>})$.

A count-based unigram LM is a simple model but is suitable given our empirical observations on word-level errors (described earlier in this section) because (1) it explicitly models word frequency, (2) it is straightforward to update as the pseudo annotated dataset improves over self-training iterations, and (3) it can be expressed as a weighted finite-state automaton which, as we discuss next, has several properties useful for our decoding method.

4.3.2 Weighted Finite State Automaton

A weighted finite-state automaton (WFSA) is a set of states and transitions between the states. Each transition accepts a particular symbol as input and has a weight associated with it. The symbols come from a finite alphabet Σ. A sequence of consecutive transitions is referred to as a "path", and the label of a path is the concatenation of all symbols consumed by its constituent transitions. The WFSA has a start state and a set of final states. A successful path is a path from the start state to a final state, and a sequence of symbols is "accepted" by the WFSA if there exists a successful path that consumes this sequence (Mohri et al., 2002).

Since we are focused on decoding and only need the best scoring path for any given sequence (i.e., Viterbi search), we consider the weights over the *tropical semiring*. That is, the weight of a path is the sum of its transition weights, and the score of a sequence of symbols is the minimum weight of all the successful paths that accept that sequence.

Decoding with the post-correction model is at the character-level (Equation 3.3), so to leverage word frequency in the decoding process, we convert the count-based word-level LM described in Section 4.3.1 to a WFSA representation that consumes and scores sequences at the character-level.

The WFSA is constructed to accept the words known to the LM by consuming each character in the word (in sequence) as input. The score of the path that accepts a known word w is the negative log of its probability from the LM: $-\log p_{\text{word}}(w)$. A simple example is shown in Figure 4.1(a).

The WFSA, as described above, can only accept a single word. However, the input and corresponding predictions of the post-correction model are sequences of words, typically lines or sentences. To enable the WFSA to accept such sequences, we add transitions that accept a set $\mathcal{B}$ of word boundary symbols (whitespace, punctuation, and end-of-sequence) from the states at the end of the known words (e.g., states 1 and 2 in Figure 4.1(a)) back to the start state. Once in the start state, the model can begin consuming characters from the next word.

Further, we modify the WFSA such that the start state is also the only final (accepting) state since the predicted sequence is considered complete only when the model predicts an end-of-sequence symbol after the last character.

Character LM for Unknown Words To enable the prediction of words unknown to the count-based LM, we include an unknown word state in the WFSA as shown in Figure 4.1(a). We add an ϵ-transition (a transition that consumes no input), with an associated cost $-\log p_{\text{word}}(\text{<unk>})$ (i.e., the probability mass reserved for unknown words in Section 4.3.1) to enter the unknown word state from the start state. The model remains in the unknown state until a word boundary symbol from the set $\mathcal{B}$ is consumed to return to the start state.

The unknown word state is designed to accept any combination of the symbols in Σ, thereby permitting the prediction of words unseen by the word-level LM. To score each character consumed at the unknown word state, we use a character-level n-gram language model.[5] We denote the probabilities from this character n-gram LM as p_{char}. The probability distribution is estimated with modified Kneser-Ney smoothing on character n-grams from unique word forms in the set $\{f_\theta(\boldsymbol{x}) \mid \boldsymbol{x} \in U\}$. We use unique word forms because unknown words are likely rare, and using count-based word forms would undesirably shift the probability mass towards more frequent words.

Thus, we are able to leverage the benefits of the word-level model on “known words" and the character-level model to score “unknown words" to influence the post-correction model to predict frequent known words while accounting for cases where there may be many unknown words (such as if the language has rich morphology).

4.3.3 Efficient scoring with the WFSA

The constructed WFSA has states to score character sequences that form known words and an unknown word state that relies on a character n-gram LM to score unknown sequences.

During inference, we independently score the next character through the known word model and the unknown word model and then choose the best scoring path. This formulation has two advantages: (1) separate scoring allows us to compactly represent the WFSA states for known words and (2) instead of representing the character n-gram LM directly in the WFSA, leading to the number of states exponentially increasing with n, we can use highly-optimized LM toolkits such as KenLM (Heafield et al., 2013) for scoring unknown words.

Known Word Model Consider the WFSA with only known word states. We apply standard algorithms for determinization and minimization on these states, which leads to an efficient and compact representation of the count-based language model (Mohri, 1996). As shown in Figure 4.1(b), the resultant minimized WFSA has several properties useful for our decoding method, discussed below.

Determinization ensures that each state has at most one outgoing transition that consumes a given input symbol, and **minimization** eliminates redundant states and transitions, reducing the time and space needed to process an input sequence.

Further, minimization includes pushing the transition weights toward the start state of the

[5] We use $n = 6$. We experimented with different values of n in early experiments but found that $n = 6$ gave us the best results for all languages in our dataset.

WFSA as much as possible (Mohri et al., 2002). This lends itself well to our method since inference in the OCR post-correction model is performed with beam search; if the cost of a path is established closer to the start state, unfavorable hypotheses can be pruned at an earlier timestep, which allows us to avoid search errors more effectively within an approximate search algorithm like beam search.

Lastly, since each state in the WFSA has at most one outgoing transition for each symbol, the transition scores can be precomputed and stored as a matrix, allowing efficient retrieval during decoding.

At decoding timestep t, let the previous timestep score from the known word model be $\text{known}(y_{t-1})$ and the current WFSA state be s_{t-1}. The score for predicting the next character y_t is the weight of the transition from state s_{t-1} that consumes y_t in the minimized WFSA (see Figure 4.1(b)). Thus,

$$\text{known}(y_t) = \text{known}(y_{t-1}) + \text{score}_{\text{wfsa}}(y_t \mid s_{t-1})$$

where $\text{known}(y_0) = 0$. If y_t does not continue the path of any known word, then $\text{score}_{\text{wfsa}}(y_t)$ is $\inf$.

Unknown Word Model We use the probability p_{char} from the character n-gram language model to score unknown words. In general, at decoding timestep t, the unknown model score for y_t will be:

$$\text{unk}(y_t) = \text{unk}(y_{t-1}) - \log p_{\text{char}}(y_t \mid y_{t-1}, \ldots, y_{t-n})$$

However, if $y_{t-1} \in \mathcal{B}$ (i.e., the previous word is complete) or $t = 0$, the WFSA is currently in the start state. To begin an unknown word, we also need to add the weight of entering the unknown word state to $\text{unk}(y_t)$, i.e., $-\log p_{\text{word}}(\text{<unk>})$.

Best Scoring Path The scores are in the tropical semiring (negative log probabilities). At timestep t, the best score for y_t from the lexical models is:

$$\text{score}_{\text{lex}}(y_t) = \min(\text{known}(y_t), \text{unk}(y_t)) \tag{4.1}$$

During decoding, we keep track of both the known and unknown model scores for the *current word* being generated in the hypothesis. When the word is completed (when $y_t \in \mathcal{B}$), both the known and unknown word models return to the start state of the WFSA (see Figure 4.1). Since the two paths are in the same state and are thus indistinguishable with respect to future predictions in the hypothesis, we choose the best scoring path to continue decoding. This is known as hypothesis recombination.

The WFSA framework, thus, allows us to efficiently represent the word-level LM in a manner that scores symbols at the character-level and accounts for unknown words. This enables joint inference with the character-level LSTM decoder in the OCR post-correction model, as discussed below.

4.3.4 Joint Decoding with the LSTM

At decoding timestep t, let $p_{\text{lstm}}(y_t)$ be the probability of generating a character y_t based on the LSTM decoder's hidden state (Equation 3.3). We also compute $\text{score}_{\text{lex}}(y_t)$, which is a negative log probability, as defined in Equation 4.1. The final probability of predicting y_t is obtained through linear interpolation between these two scores,[6] weighted by a hyperparameter λ:

$$p(y_t) = (1 - \lambda) \cdot p_{\text{lstm}}(y_t) + \lambda \cdot p_{\text{lex}}(y_t) \tag{4.2}$$

where $p_{\text{lex}}(y_t) = \exp(-\text{score}_{\text{lex}}(y_t))$.

This joint decoding strategy is applied when performing inference with beam search using a trained OCR post-correction model. When used in combination with self-training, the predictions made by the model improve as we repeat the self-training process, iteratively improving the count-based LM and resulting in a better distribution of $p_{\text{lex}}(y_t)$.

4.4 Experiments

In this section, we present experiments with our semi-supervised post-correction method on the four typologically diverse endangered languages in our dataset.

4.4.1 Experimental Setup

Datasets We use the OCR post-correction dataset presented in Chapter 2 which contains transcribed documents in four endangered languages: Ainu, Griko, Kwak'wala, and Yakkha. Recall that because of the effort involved, we manually annotated only a small subset of the pages in the documents. The scanned images of the remaining pages are used for semi-supervised learning. The sizes of the annotated and unannotated sets are described below:

[6] We leave other interpolation techniques like log-linear interpolation as potential future work.

- **Ainu** (ain): The dataset contains 816 manually transcribed lines and 7,646 lines without gold transcriptions.
- **Griko** (grk): The dataset contains 807 and 3,084 sentences with and without gold transcriptions, respectively.
- **Yakkha** (ybh): In total, there are 159 manually transcribed sentences and no unannotated lines in the dataset. Therefore, as the unannotated set, we use the first pass OCR on the validation and test sets in a transductive learning setting (≈ 30 sentences: see below for data splits).
- **Kwak'wala** (kwk): The dataset contains 262 gold-transcribed lines and 2,255 unannotated lines.

Data Splits We perform 10-fold cross-validation for all experiments, following the same splits as those in Chapter 3. For the semi-supervised learning, we use the same set of unannotated images across all cross-validation folds (except in the case of Yakkha, see the unannotated set description above).

Metrics We evaluate our systems in terms of character error rate (CER) and word error rate (WER), both standard metrics for measuring OCR and OCR post-correction performance, as described in Section 2.3.

Methods In our experiments, we compare the performance of the following methods:

- First-Pass: This is the performance of the first pass OCR system. From our analysis in Chapter 2, we choose the best performing first pass system, the details of which are in Section 2.4. We use Ocular for Kwak'wala and Google Vision for Ainu, Griko, and Yakkha.
- Base: The OCR post-correction model for endangered language texts, presented in Chapter 3.
- Semi-Supervised: Our proposed method as described in Section 4.2 and Section 4.3.

Implementation The neural post-correction models are implemented using the DyNet neural network toolkit (Neubig et al., 2017). The WFSA is implemented using the MFST Python wrapper on OpenFST (Francis-Landau, 2020), and we use the KenLM toolkit (Heafield et al., 2013) to train and query the character n-gram language model. The results reported are the average of five randomly seeded runs (i.e., five runs for each of the 10 cross-validation folds).

	% Character Error Rate				% Word Error Rate			
Model	`ain`	`grk`	`ybh`	`kwk`	`ain`	`grk`	`ybh`	`kwk`
First-Pass	1.34	3.27	8.90	7.90	6.27	15.63	31.64	38.22
Base	0.80	1.70	8.44	4.97	5.19	7.51	21.33	27.65
Semi-Supervised								
Self-Training	0.82	1.45	7.20	4.00	5.31	6.47	18.09	23.98
Lexical Decoding	0.81	1.51	7.56	4.28	5.18	6.60	19.13	25.09
Both	**0.63**	**1.37**	**5.98**	**3.82**	**4.43**	**6.36**	**16.65**	**22.61**
Error Reduction $\left(\frac{\text{Base}-\text{Both}}{\text{Base}}\right)$	21%	19%	29%	23%	15%	15%	22%	18%

Table 4.1: Our semi-supervised approach improves performance over the baselines (10-fold cross-validation averaged over five randomly seeded runs). "Self-Training" and "Lexical Decoding" refer to experiments where we use these methods independently. "Both" refers to their combination. We **highlight** the best model for each language.

4.4.2 Main Results

Table 4.1 shows the performance of the baselines and our proposed semi-supervised approaches for the four languages in the dataset. For all languages, using semi-supervised learning leads to substantial reductions in both CER and WER.

We note that we did a hyperparameter search over the number of self-training iterations and the weight of the WFSA λ, and Table 4.1 presents the best models based on the validation set WER. Extensive analysis of these factors is in Section 4.4.4.

First, we note that the Base post-correction method improves error rates over the first pass for all languages. With our proposed semi-supervised learning method, combining self-training with lexically-aware decoding leads to the best performance across all the languages, with error rate reductions in the range of 15%-29%.

This is especially noticeable in Ainu, where using either self-training or lexical decoding independently results in worse performance than the Base system, but jointly using them improves the CER by 21%. For the other languages, the independent components improve over the base model but less so than their combination. This indicates the complementary nature of the two components: the language model used for lexically-aware decoding is improved by self-

Known Word	Unknown Word	% Character Error Rate				% Word Error Rate			
Model	Model	`ain`	`grk`	`ybh`	`kwk`	`ain`	`grk`	`ybh`	`kwk`
CharLM	*(not needed)*	0.64	1.43	6.22	3.85	4.50	6.44	16.78	22.90
WordLM	Character uniform	0.64	1.42	6.12	3.95	4.50	6.39	16.71	23.11
Ours	Character n-gram	**0.63**	**1.37**	**5.98**	**3.82**	**4.43**	**6.36**	**16.65**	**22.61**

Table 4.2: A more informed unknown word model (character n-gram) in combination with the word-level known word model consistently performs better than the alternatives for all four languages in our dataset.

training. In turn, it reinforces correctly predicted words to counteract the influence of incorrect pseudo-annotated instances.

4.4.3 Comparing Language Models

Our proposed decoding method uses a count-based word-level LM in combination with a character n-gram LM to compute p_{lex} for joint decoding with the LSTM decoder (Equation 4.2). In this section, we substitute this model with two other variants of count-based LMs to compute p_{lex}:

- CharLM: We use a character 6-gram language model on the model predictions from self-training $\{f_\theta(\boldsymbol{x}) \mid \boldsymbol{x} \in U\}$, estimated with modified Kneser-Ney smoothing.
- WordLM: We use the word-level LM described in Section 4.3.1, but do not use a character n-gram model for unknown words. Instead, we score unknown words with a simple uniform probability over all characters in the vocabulary.

We tune λ on the validation set for each model independently and report results with the best setting in Table 4.2. Using either CharLM or WordLM for lexically-aware decoding improves the error rates with respect to the Base model. The word-level model performs better for all languages except Kwak'wala, likely due to the large percentage of unknown words in this language. We also see that our proposed method, which leverages a count-based word-level LM for known words combined with a character-level LM for scoring unknown words, results in the best performance overall.

Although not observed in our dataset, we note that some printed materials have a high degree of spelling variation or contain texts for which word tokenization is difficult. In such

Lang.	LM	Known		Unknown	
Code	Coverage	Base	Ours	Base	Ours
ain	0.97	0.95	0.98	0.08	0.25
grk	0.94	0.89	0.96	0.51	0.71
ybh	0.68	0.90	0.95	0.51	0.59
kwk	0.59	0.89	0.92	0.50	0.58
Average	0.80	0.91	**0.95**	0.40	**0.53**

Table 4.3: Our method improves over the base model on words that are both known and unknown to the WFSA. We show the fraction of known test words, and the fraction of correctly predicted known and unknown words.

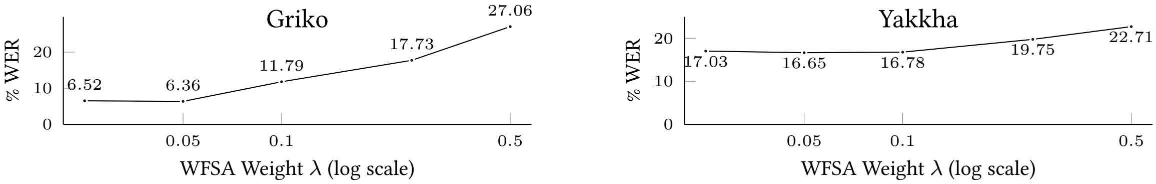


Figure 4.2: The weight of the WFSA during joint decoding can affect word error rate (sometimes significantly, as in Griko; top). All other hyperparameters are kept equal and correspond to the best systems in each language.

cases, the word-level model may not be as effective, but CharLM can still be used with the proposed lexically-aware decoding framework to obtain improved performance over the baseline method.

4.4.4 Analysis

We analyze specific components of our model to understand the advantages of our proposed approach.

Known vs. Unknown Words We first identify the source of the improvements that our approach makes over the baseline. Table 4.3 presents the fraction of correctly predicted words, split on whether these words are “known" to the WFSA (i.e., in the vocabulary of the word-level

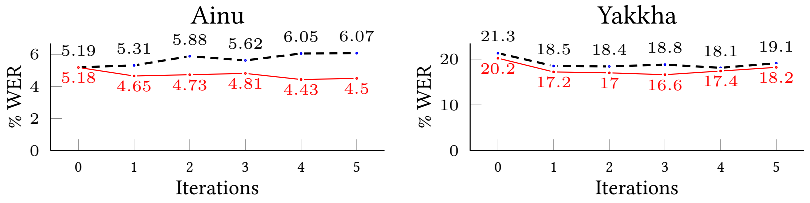


Figure 4.3: Integrating lexically-aware decoding through interpolation with a WFSA (red lines) aids self-training in improving WER across iterations. Black dashed lines correspond to self-training without lexical decoding.

LM) or “unknown". Intuitively, we expect that decoding with the WFSA will improve prediction on the known words.

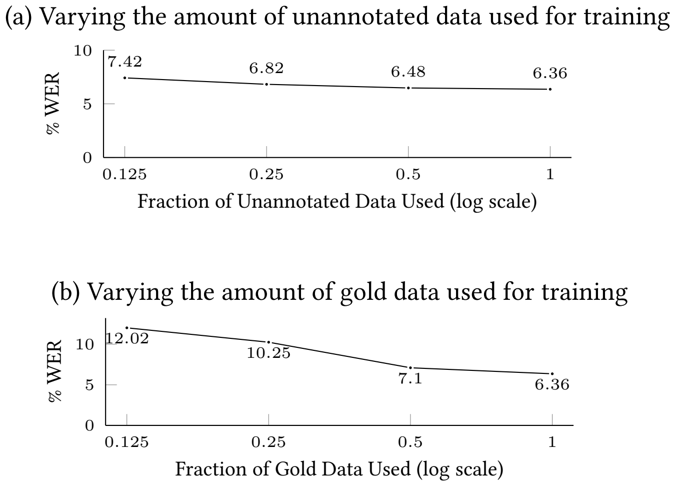
Compared to the baseline, our method improves on words known to the WFSA, moving from 91% to 95% accuracy on average. Our method also improves unknown word prediction over the baseline from an average accuracy of 40% to 53%. In cases like Kwak’wala, where, due to the rich morphology of the language, more than 40% of the test words are unseen, including an unknown word model in the WFSA is particularly important.

Effect of WFSA Weight One of the important hyperparameters of our lexically-aware method is the weight that we place on the WFSA score during inference (λ in Equation 4.2). Specifically, in the case of Griko, we find that the value of this hyperparameter can significantly affect performance. As shown in Figure 4.2, high weights of λ (i.e., more weight on the WFSA) lead to suboptimal WER, while lower λ leads to much better performance.

This hyperparameter is less important in the other three languages, leading to smaller variations in performance. As an example, we depict the effect on Yakkha in Figure 4.2, where increasing λ does not affect performance as much as in Griko.

Self-Training Iterations The evolution of WER across 5 self-training iterations for Ainu and Yakkha is shown in Figure 4.3. Particularly for Ainu, we see that combination with lexically-aware decoding is crucial for the success of self-training. For Yakkha, self-training does improve performance independently but is more effective when lexically-aware decoding is used (error rates on Griko and Kwak’wala follow a similar trend).

Dataset Size We study the effect of varying the amount of gold-transcribed and unannotated data used for training. The WER when varying the size of the Griko datasets is shown in


(a) Varying the amount of unannotated data used for training

(b) Varying the amount of gold data used for training

Figure 4.4: Even a small amount of unannotated data is useful for our semi-supervised method, improving WER over Base (WER=7.51) in (a). Varying the size of gold-annotated data has a stronger effect on post-correction performance in (b). Results are shown with Griko.

Figure 4.4 (the size of each set is varied while keeping the other set at its full size). We see that reducing the amount of gold-transcribed data worsens WER significantly. On the other hand, reducing the unannotated data has a smaller effect: even with a little unannotated data, our method improves over the Base model.

Error Rate in the First Pass OCR To evaluate how the error rate in the first pass OCR transcription affects subsequent post-correction, we measure the performance of our proposed method when applied to first pass outputs from two OCR systems: Google Vision and Ocular (described in Section 2.4). Figure 4.5 shows the WER on the Kwak'wala dataset. We see that, although Google Vision has a much higher first pass error rate than Ocular, the post-correction model improves performance over both OCR systems. We also note that the relative error reduction is higher for the Google Vision system (68%) than for Ocular (41%), likely because the Ocular LM is trained on the same data as the post-correction model.

Qualitative Analysis In Figure 4.6, we show examples of errors fixed as well as errors introduced by our post-correction model as compared to the baseline system. In Figure 4.6 (a) and (b), we see that although the baseline corrects some of the errors in the first pass OCR, it also introduces errors such as extra diacritics and incorrect substitutions. Using our proposed method leads to an error-free transcription of these images. However, in Figure 4.6 (c) and (d),

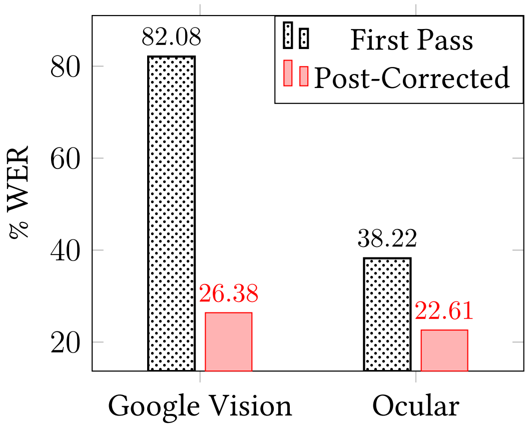


Figure 4.5: Our post-correction model significantly improves recognition accuracy over different first pass OCR systems that have varied error rates (Google Vision and Ocular). Results are shown with Kwak'wala.

we see that our method occasionally introduces errors in predictions. Specifically, although the model fixes the first pass errors, it generates words that are considerably different from the target. Such errors likely occur when the model follows an incorrect path in the WFSA during lexically-aware decoding. Since we are using beam search, the correct path cannot be recovered if it was pruned at an earlier timestep.

4.5 Related Work

While existing neural post-correction methods (Dong and Smith, 2018; Rigaud et al., 2019; Hämäläinen and Hengchen, 2019), as well as the method presented in Chapter 3, do not rely on lexical information, some earlier methods use dictionaries to improve performance. For example, Tong and Evans (1996) and Niklas (2010) use lexicons in combination with n-gram context to generate post-correction candidates for erroneous words. These methods are typically evaluated on English and assume the presence of high-coverage lexicons (Schulz and Kuhn, 2017), making them difficult to adapt to endangered languages.

Related to our decoding method are models that incorporate lexical knowledge into neural machine translation models. Arthur et al. (2016) propose adding a dictionary for translating low-frequency words and Zhang et al. (2018) improve decoding by upweighting translations that contain relevant words. Additionally, there are methods which add *hard* lexical constraints by forcing predictions to contain user-specified words and phrases (Hokamp and Liu, 2017; Post

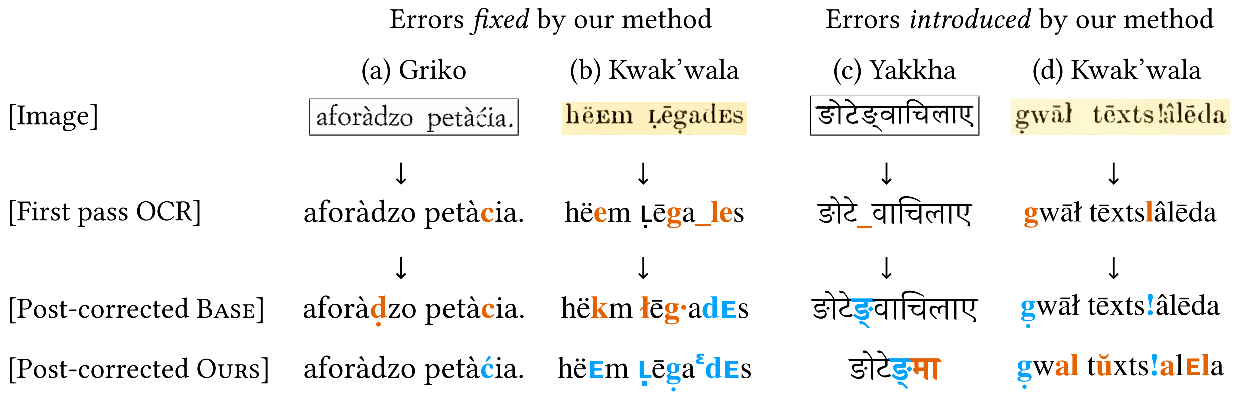


Figure 4.6: Our post-correction model **fixes** many of the first pass OCR **errors** that the base model does not fix such as (a) and (b). In rare cases, our method introduces **errors** into the transcription such as (c) and (d).

and Vilar, 2018).

Lastly, we note that our proposed approach combines information from a neural model and a finite-state machine to leverage the advantages of both. In a similar direction, Rastogi et al. (2016) and Lin et al. (2019) design finite state architectures with paths weighted by contextual features from an LSTM. These methods use joint parameterizations of the models and are thus more complex to train (particularly in the low-resource setting) than the joint decoding method we present in this chapter.

4.6 Summary

In this chapter, we present a semi-supervised learning technique for OCR post-correction that improves performance over our previous model without any additional manual annotation. The semi-supervised algorithm has two essential components: (1) self-training, which uses unlabeled data to create pseudo-training data with a baseline model, and (2) lexically-aware decoding, which derives a lexicon automatically from the self-training data and uses word frequency information in a WFSA framework to enforce consistency in the model's predictions. We also analyze how each of the two components affects learning, and find that they are complementary, where their combination results in the best performance across all languages in our dataset. Compared to the model described in Chapter 3, using semi-supervised learning improves character error rates by an average of 23% and word error rates by an average of 17.5% over the languages in our dataset. Overall, our best models reduce character error rates by 49% and word error rates at 44% on average over existing OCR systems from which we obtain first-pass transcriptions.

Finally, we extensively evaluate how the performance of the model is affected by dataset characteristics, including the fraction of unknown words and the sizes of the labeled and unlabeled sets; modeling factors such as the unknown word model architecture and the linear interpolation weight of the WFSA in decoding; and the OCR system used for the first-pass.

Chapter 5

Case Study: Impact of OCR on Kwak’wala Language Revitalization

In Chapter 3 and Chapter 4, we described models for post-correction that improve error rates of OCR transcriptions, even in very low-resource settings. In this chapter, we look beyond error rates and attempt to evaluate the impact the models proposed in this thesis could have on linguistic research, language documentation and revitalization, and communities that speak endangered languages. We focus on the Kwak’wala language as a case study to highlight the potential benefits of improving OCR for endangered languages.

Kwak’wala is a member of the Wakashan language family spoken on the Northwest North American Coast. Heritage learners and teachers of five dialects from 18 Kwaka̱kwa̱’wakw Nations are actively engaged in the revitalization of spoken Kwak’wala. Written documentation of the language extends back over 120 years (Boas, 1897; Boas and Hunt, 1902; Boas, 1911; Boas and Hunt, 1921; Boas, 1934, *inter alia*). The knowledge held in these texts has tremendous value for community-based research, but to the extent they have been digitized, they are still ‘trapped’ in image files that are not machine-readable and are written in a technical and somewhat idiosyncratic orthography primarily used in archival research contexts.

As discussed in earlier chapters, the orthography used in the documentation produced by Hunt and Boas is complex and uses many unique diacritics and digraphs (Boas, 1900) that existing OCR systems frequently recognize incorrectly. In the case study presented in this chapter, we ‘unlock’ these resources using our OCR post-correction model for Kwak’wala, converting hundreds of scanned images from these documents into machine-readable text, thus enabling much easier access to community members. Additionally, we conduct a user study to evaluate the downstream utility of our models. The study measures how much time a human transcriber needs to spend on *correcting the outputs from OCR systems* to obtain an accurate transcription, and we quantitatively analyze whether our proposed post-correction method has utility in reducing the time (and thus effort) spent on manual correction. Finally, we attempt to make the extracted texts more readable to Kwak'wala researchers, speakers, and language learners by automatically transliterating them from the legacy Hunt/Boas orthography into the more modern U'mista writing system (Nicolson and Werle, 2009), which is currently a community-preferred orthography. Some of the material in this chapter was also presented in Rosenblum et al. (2022).

5.1 Documents in the Kwak'wala Language

Communities that speak Kwak'wala have a long tradition of written documentation about the language and the culture. Research, resource creation, and knowledge continuity among community members are robust activities among 18 Kwaka̱kwa̱'wakw Nations, supported by active community engagement. Language teaching and learning are a focus in many communities, intertwined with the reclamation of both culture and territorial sovereignty.

We focus on creating machine-readable text resources from a collection of documents produced by anthropologist Franz Boas in collaboration with George Hunt, a native speaker of Kwak'wala. Franz Boas first met George Hunt in 1887, and the long collaboration between Hunt and Boas which followed focused primarily on documenting the Kwaka̱kwa̱'wakw culture (Berman, 1994). The documentation was written in the Kwak'wala language. Over four decades, Hunt and Boas generated an extensive collection of linguistic and cultural documentation: 14 published bilingual Kwak'wala–English texts with over 3000 pages in total, an unpublished dictionary, and additional unpublished manuscripts now held in three archival repositories in Philadelphia and New York. These texts encompass a grammar of the language; word lists; stories; recipes; procedural texts; descriptions of practices, beliefs, and customs; names of plants and descriptions of their uses; calendars and lists of month names from various communities; descriptions of dialectal differences; maps and lists of placenames; and more. An example page from the published volumes we extract text from is in Figure 5.6.

For Kwaka̱kwa̱'wakw communities today, these texts are rich troves containing precious knowledge, of great interest to community members, and special value to community-led projects focused on teaching, learning, strengthening, and reclaiming their language, cultural practices, and territorial sovereignty (Lawson, 2004). Even though they are published documents in the public domain, layers of obstacles impede access and limit the reach of these resources: for example, manuscripts in the Columbia University Rare Books collection have not been scanned,

U’mista	a	a̱	b	d	dł	dz	e	g	gw	g̱
Boas	a, ā	ᴇ, ă, î, ŭ	b	d	ʟ̣	dz	ä, ê	g·	gw, gᵘ	g̣
IPA	a	ə, a, ɪ, ʊ	b	d	dl	dz	ɛ, e	gʲ	gʷ	ɢ

U’mista	g̱w	h	i	k	kw	k̓	k̓w	ḵ	ḵw	ḵ̓
Boas	g̣w, g̣ᵘ	h	i, ī, e, ē, ë	k·	kw, kᵘ	k·!	k!w, k!ᵘ	q	qw, qᵘ	q!
IPA	ɢʷ	h	i, e	kʲ	kʷ	k’ʲ	k’ʷ	q	qʷ	q’

U’mista	ḵ̓w	l	’l	ł	m	’m	n	’n	o	p	p̓	s	t	t̓	tł
Boas	q!w, q!ᵘ	l	ᵋl̇	ł	m	ᵋm	n	ᵋn	â, ô	p	p!	s	t	t!	ʟ
IPA	q’ʷ	l	l’	ɬ	m	m’	n	n’	ɔ, o	p	p’	s	t	t’	tł

U’mista	t̓ł	ts	t̓s	u	w	’w	x	xw	x̱	x̱w	y	’y	’
Boas	ʟ!	ts	ts!	u, ū, o, ō	w	ᵋw	x·	x̣w, x̣ᵘ	x	xw, xᵘ	y	ᵋy	ᵋ
IPA	t’ɬ	ts	t’s	u, o	w	w’	xʲ	xʷ	χ	χʷ	j	j’	ʔ

Figure 5.1: Transliteration of alphabet sequences between the U’mista and Boas orthographies for Kwak’wala along with their representation in the international phonetic alphabet (IPA). The U’mista system is modern and one of the currently preferred orthographies in the community. However, large collections of cultural and linguistic documentation are written in the older Boas orthography which, as the figure shows, is more complex with the use of unique diacritics and digraphs. Figure adapted from Nicolson and Werle (2009).

and can only be consulted by scheduling a visit to the archive. Scanned images of the published volumes are accessible online in PDF format through the Smithsonian,[1] and such digital scans allow remote access by those who are not able to travel to Philadelphia or New York. However, because the text in images is not machine-readable, it is not searchable and researchers potentially need to look at tens or hundreds of images to locate relevant information.

Developing reliable OCR to extract these texts into a machine-readable format can serve the community in many ways. Their contents can be strategically tagged, selected, indexed, rearranged, reformatted, excerpted, and adapted according to various needs and preferences. Additionally, the writing system used by Boas and Hunt is radically different from the orthographies the community currently prefers. Figure 5.1 shows the transliteration between U’mista, a community-preferred orthography, and the more complex Hunt/Boas writing system. Many researchers draw on the Hunt/Boas materials in their work and integrate this knowledge into their community-based language and culture work, but developing comfort with the writing system and the texts requires a significant investment of time and energy, and sharing infor-

[1] https://library.si.edu/digital-library/book/annualreportofbu351smit

mation and knowledge from these materials with others often requires retyping excerpts into a different writing system in order to make content legible to a broader audience. Converting the images to machine-readable text opens the door to automatic transliteration from one orthography to another, enabling wider access to the valuable information contained in these documents. Beyond search and transliteration, extracting the text with OCR will also enable dataset creation for the Kwak'wala language and lay the groundwork for other NLP applications that the community is interested in developing, including a language model and an automatic speech-to-text transcription system.

Focusing on improving OCR for the Hunt/Boas publications thus offers significant utility and impact for community language revitalization efforts in a multitude of ways: by improving accessibility to these culturally important documents through search and transliteration as well as by increasing the amount of Kwak'wala language data available for downstream NLP tasks.

Improving OCR for the Hunt/Boas Orthography We use our proposed post-correction model to improve OCR performance beyond existing systems. As presented in Chapter 4, the best performing model for Kwak'wala uses the semi-supervised learning technique that combines self-training and lexically-aware decoding. Compared with the first-pass OCR system (Ocular), post-correction reduces character error rate by 52% and word error rate by 41% on Kwak'wala documents that are written in the Hunt/Boas orthography (see Table 4.1 for detailed results). We apply the trained model on two full volumes of the Hunt/Boas publications, converting over 1,500 pages to machine-readable text. A full description of the resources we create and publicly release is in Section 5.3.

Although the character and word error rate metrics are useful for understanding OCR performance, our goal is to provide transcriptions that are accurate and useful to Kwak'wala community members and researchers. In the following section, we take a human-centered approach in evaluating our OCR pipeline to understand whether the automatically produced transcriptions are beneficial to downstream users that access the information in these documents.

5.2 Evaluation with a User Study

Traditionally, accurate transcriptions of scanned documents like the Hunt/Boas publications would be produced by a human annotator, who would look at the scanned image of each document and type out the text present in it – a process that is time-consuming and requires significant manual effort. To evaluate the utility of the outputs from our models, we analyze whether using OCR and OCR post-correction before the manual transcription process is effective in reducing the time spent by a human annotator in producing an accurate transcription. We conduct a user study where we compare the time spent by transcribers on producing an accurate transcription in various settings with and without the use of an OCR system. We attempt to answer two primary questions:

1. Is it faster for a human transcriber to correct the errors in an OCR output as compared to typing out the text from scratch?
2. Does adding the post-correction model affect transcription speed beyond existing off-the-shelf OCR tools?

We design controlled experiments to measure human transcription speed on a subset of images from the Hunt/Boas texts and evaluate how transcription time is affected in various settings to understand whether there is utility in introducing OCR into the process. Additionally, we obtain subjective feedback on how having OCR outputs affected the transcription task through a survey sent to participating transcribers after tasks were completed.

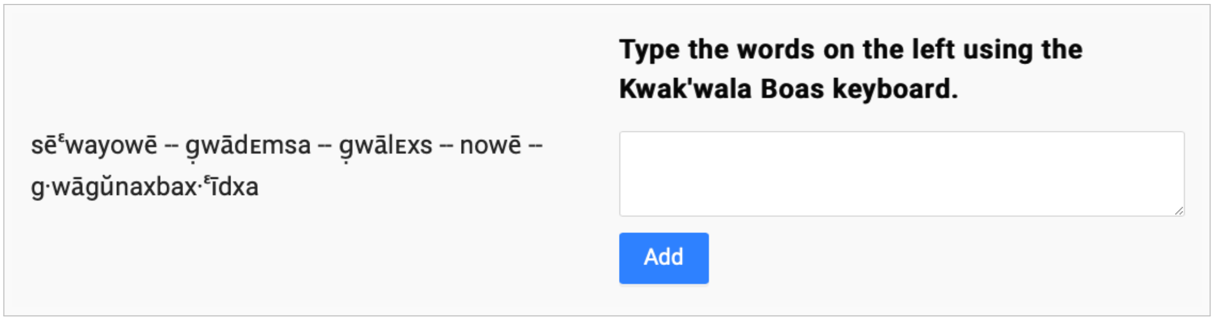
5.2.1 Participants

We employed nine participants for the user study, all of whom had some transcription experience. Of the nine, two participants had familiarity with the Kwak'wala language as well as the Hunt/Boas texts and the orthography – one is a heritage Kwak'wala language learner and the other is an academic linguist working with Kwak'wala language materials.

We also employed seven participants that had no experience or familiarity with Kwak'wala. Three of these participants are graduate students at a university in the United States and four participants were employed through Upwork,[2] a marketplace for freelance professionals. We selected them based on prior transcription experience, knowledge about data annotation for machine learning, and linguistic training as well as a high job success rate on the Upwork platform.[3] Including participants with varying degrees of prior knowledge of the Kwak'wala language also allowed us to evaluate whether this is a factor that affects transcription speed and the overall experience with the user study tasks.

[2] https://www.upwork.com/

[3] Full IRB approval was obtained for the user study, all participants signed a consent form before working on the transcription tasks, and all data collected was anonymized.



sēᵋwayowē -- g̣wādɛmsa -- g̣wālɛxs -- nowē --
g·wāgŭnaxbax·ᵋīdxa

Type the words on the left using the Kwak'wala Boas keyboard.

Figure 5.2: Practice task for transcribers to become familiar with the Boas keyboard. We included eight practice tasks in the Label Studio interface to cover all special character combinations in the Boas orthography multiple times. Users could repeat tasks as many times as they wanted to before moving on to the main transcription task.

5.2.2 Transcription Interface and Keyboard

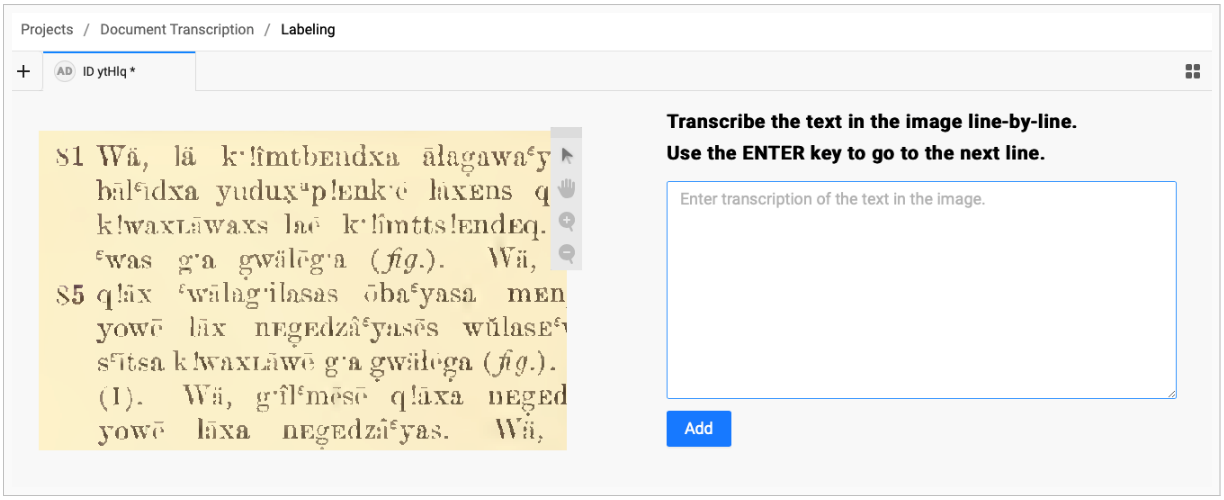
We use Label Studio,[4] an open-source data annotation interface for setting up transcription tasks for the user study. We customized the interface for the transcription task and additionally modified it to record information necessary for our analysis of transcription speed, including timestamps for when transcribers operate on each task.

Many characters and diacritics in the Hunt/Boas orthography are not present on a standard computer keyboard. To increase transcription efficiency, we used Keyman Developer[5] (an open-source toolkit) to create a keyboard for representing the characters in the orthography. The keyboard maps standard US English keyboard keystrokes to characters in the Hunt/Boas orthography. A detailed description of the keyboard layout and usage is in Section A.1. All participants were required to use this virtual keyboard to ensure consistency in terms of typing efficiency across all transcribers.

To train participants before the user study experiments, we designed a keyboard practice task, which presents a few sentences of text in the Hunt/Boas orthography that the transcriber has to type using the keyboard. The practice texts were selected such that all the different diacritic and digraph keystroke combinations were covered multiple times. The practice tasks were also added to the Label Studio web interface – a screenshot of the interface for the practice task is shown in Figure 5.2. Participants were able to repeat the practice tasks as many times as needed to gain familiarity with the keyboard. Additionally, we added keystroke mapping

[4] https://labelstud.io

[5] https://keyman.com/developer/


Projects / Document Transcription / Labeling
+
AD ID ytHlq *
Enter transcription of the text in the image.
Add

81 Wä, lä k·!îmtbᴇndxa ālagawaᵋy
bālᵋidxa yudux̣ᵘp!ᴇnk·ē lāxᴇns q
k!waxʟāwaxs laē k·!îmtts!ᴇndᴇq.
ᵋwas g·a gwälēg·a (*fig.*). Wä,
85 q!āx ᵋwālag·ilasas ōbaᵋyasa mᴇn
yowē lāx ɴᴇgᴇdzâᵋyasēs wŭlasᴇᵋ
sᵋītsa k!waxʟāwē g·a gwälēga (*fig.*).
(1). Wä, g·îlᵋmēsē q!āxa ɴᴇgᴇd
yowē lāxa ɴᴇgᴇdzâᵋyas. Wä,

Transcribe the text in the image line-by-line.
Use the ENTER key to go to the next line.

Figure 5.3: Transcription task interface, designed in Label Studio. The interface displays the image of a page and a text box to enter the transcription into. It also has zoom and pan tools for the image, allowing users to zoom in on characters that might be hard to identify. The figure depicts a cropped image for clarity. When an OCR system or our post-correction method is used before the manual transcription task, the text box on the right is pre-filled with the output transcription from the model and the user's task is to correct any remaining errors.

information to the interface for all tasks (transcription and practice tasks) for users to quickly reference.

5.2.3 Transcription Task Settings

The primary objective for the participants was to produce an accurate transcription of the image presented to them in each task. In the Label Studio interface, as seen in Figure 5.3, the image is displayed alongside a text box for the user to enter the transcription. To evaluate whether using OCR is useful in reducing transcription speed, we have three different setups for the tasks:

- **Baseline**: This setup does not include the use of any OCR system. The transcriber must type out the text seen in the image from scratch – they are presented with the image and an empty text box in the interface (see Figure 5.3). This setup represents our baseline for measuring transcription speed.
- **Off-the-shelf OCR**: In this setup, we use an off-the-shelf OCR tool (Ocular) on the image for each task prior to the manual annotation. The transcriber is presented with the image and a text box containing the OCR output – that is, the text box on the right in Figure 5.3

A	B	C	D
B	C	D	A
C	D	A	B
D	A	B	C

A	B	C	D
B	A	D	C
C	D	B	A
D	C	A	B

Figure 5.4: Two examples of 4x4 Latin Squares. Each symbol appears only once in each row and each column. The number of symbols is the same as the number of rows and columns in the square. Figure adapted from Dean and Voss (1999).

will be pre-filled with the OCR output. The task here involves looking at the text present in the image (which is the target text) and editing the OCR output in the text box to correct all the errors and produce an accurate transcription.

- **Post-correction**: This is similar to the previous setup, but we use a pipeline that includes applying our OCR post-correction method on the output from the off-the-shelf tool. The transcriber is presented with the image alongside a text box containing the post-corrected transcription, and the task is to correct any remaining errors.

5.2.4 Experiment Design

While measuring transcription speed for a single page is relatively straightforward, determining whether there is a statistically significant difference in speed between the three different setups described above requires consideration of several factors. For example, a single transcriber cannot be assigned the same page multiple times with different setups as they would become familiar with the page's content, potentially leading to incorrect estimation of speed differences. Additionally, some participants may be faster at transcription in general and some pages in the document may be more challenging than others – these factors need to be accounted for when measuring transcription time across the task setups.

In statistics, such factors are known as sources of variability (or nuisance factors). We design the transcription tasks to control the variability introduced by these factors using the Latin Square Design (Dean and Voss, 1999) to assign tasks to each transcriber. The Latin Square has the same number of rows and columns (square-shaped), with a specific symbol appearing exactly once in each row and exactly once in each column; Figure 5.4 shows two examples of a Latin Square design that has 4 rows and 4 columns. This design allows control of two sources of variability – one along the rows and one along the columns.

	page1	page2	page3
user1	ocr	base	post
user2	post	ocr	base
user3	base	post	ocr

	page4	page5	page6
user1	ocr	post	base
user2	base	ocr	post
user3	post	base	ocr

	page7	page8	page9
user1	post	ocr	base
user2	base	post	ocr
user3	ocr	base	post

Figure 5.5: Task setup assignments for a group of three users using the Latin Square design. We use 3x3 Latin Squares because we have three task setups: Baseline (*base*), off-the-shelf OCR (*ocr*), and post-correction (*post*). We need three squares for each group of users because we have nine pages for transcription. All users transcribe the same set of pages, but with the Latin Square framework, they have different task setups for each page which helps control sources of variability. Note that all user identifiers and page identifiers are randomized before applying the Latin Square design.

Since we have three task setups, we choose a 3x3 Latin Square – each setup appears only once in each row and column. The two sources of variability we control are (1) the user doing the transcription and (2) the page being transcribed. We randomly divide the nine participants into three groups of three users each (to fit the 3x3 square) and choose a fixed set of nine pages from the documents that all participants will transcribe in their tasks. For each group of three users, we form three squares (since we have nine pages). The task setups – i.e., baseline, off-the-shelf OCR, post-correction – are randomly assigned within the Latin Square constraints. Adding randomization for all factors (user, page, task setup) is aimed at spreading out the effect of undetectable or unsuspected characteristics. An example of task setup assignments for one group of three users for the nine pages is in Figure 5.5.[6]

Therefore, each user has nine transcription tasks with the task setups evenly distributed so all users are sufficiently timed on each setup. The user does not transcribe the same page more than once, but all users transcribe the same set of nine pages (with varied task setups). The Latin Square Design, thus, introduces randomness across the factors to reduce variance and improve the generalization of the statistical analysis.

Dataset selection We selected nine pages from the Hunt/Boas volumes for the user study experiments, which were randomly chosen from a larger subset of 50 pages that community-

[6] We follow https://online.stat.psu.edu/stat503/lesson/4/4.4 and randomize Latin Squares separately for each group of users and each set of pages, so the task setup assignments may not look identical across groups.

based researchers deemed representative of the volumes and important to transcribe.

5.2.5 Evaluation Procedure

The nine transcription tasks were designed to take approximately 7 hours to complete. The participants accessed the Label Studio interface remotely through any web browser and first completed the keyboard practice tasks described above. Then, the participants began the transcription tasks and the interface recorded all timestamps for when transcriptions were edited and submitted. After the participants completed all tasks, we collected the timestamp information and computed how long it took to complete each task – with nine users transcribing nine pages each, we have 81 measurements of transcription speed to be used for quantitatively evaluating the utility of the OCR systems. We also calculated the character error rate (CER) of each transcription with respect to the transcription for the same page by our most experienced participant (a Kwak'wala heritage language learner who is very familiar with the orthography and had transcribed parts of the Hunt/Boas volumes prior to the user study), and discarded time measurements for transcriptions with CER > 1%. Across all 81 transcriptions, only one had an error rate higher than this threshold, and thus, the quantitative analysis below is conducted with 80 time measurements.

We also obtained qualitative feedback through a short survey that the participants filled out after completing the transcriptions. The survey asked several questions about the experience with the user study, including if the transcribers found specific tasks more difficult than others, whether they preferred typing from scratch or correcting OCR outputs (and which they thought was faster) as well as general feedback on the task and interface.

5.2.6 Quantitative Analysis

To quantify the effect of introducing OCR into the transcription process, we analyze the measurements of transcription speed that were collected from the user study tasks. As stated previously, we cannot use the time values directly to make a generalized conclusion because transcription time is not independent of the sources of variability. Instead, we use the statistical technique of Linear Mixed Effects (LME) modeling (Bates, 2007) to describe the relationship between the response variable (the transcription time) and the factors that contribute to variance. The term "mixed effects" refers to a combination of random effects and fixed effects. We have two random effects:

1. *transcriber identity*, which can take values from user1 to user9;

Task Setup	Transcription Time (minutes)	p-value
Baseline (no OCR)	61.65	3.04e-07 *
With OCR	−33.44	4.80e-08 *

Table 5.1: Per-page transcription time estimates from the linear mixed model comparing the baseline, which does not use any OCR, with the task setups that use some form of OCR (either off-the-shelf or post-correction). The time estimate for producing an accurate transcription of a page is reduced by 33.44 minutes when OCR technologies are used beforehand. The p-value is < 0.05 for the estimates, indicating statistical significance.

2. *page number*, which can take values from page1 to page9.

We also have two fixed effects:

1. *transcriber group*, which can either be yes or no indicating prior familiarity with the Kwak'wala language or not;
2. *task setup*, which can be one of the three setups described above – baseline, off-the-shelf OCR, or post-correction.

The LME estimation models the transcription time as a function of the above random and fixed effects. Using the estimations, our primary analysis attempts to identify whether the task setup affects transcription time in a statistically significant manner. We additionally look at whether the transcriber group (i.e., prior knowledge of Kwak'wala) plays a role in how fast the user completes tasks.

Does having some form of OCR help reduce transcription time? In Table 5.1, we present transcription time estimates from the LME model comparing two settings: (1) the baseline setup which does not use any OCR and the user types the transcription from scratch, and (2) having some form of OCR before the transcription process which the user can correct to produce error-free text (either the off-the-shelf setup or the post-correction setup). As is evident from the results, having some form of OCR greatly improves transcription speed, reducing the time estimate by over 50% and consequently, reducing the manual effort needed to produce an accurate machine-readable version of the documents.

Does post-correction help reduce transcription time beyond using an off-the-shelf OCR tool? From the previous results, it is evident that using OCR is beneficial in reducing

Task Setup	Transcription Time (minutes)	p-value
Off-the-shelf	31.67	2.55e-05 *
Post-correction	−6.69	0.0121 *

Table 5.2: Per-page transcription time estimates from the linear mixed model comparing task setups using an off-the-shelf OCR system (Ocular) with using our proposed post-correction method. The time estimate is reduced by 6.69 minutes for a page when we use post-correction, indicating the utility of our method to downstream users over existing OCR systems. The p-value is < 0.05 for the estimates, indicating statistical significance.

Transcriber Group	Transcription Time (minutes)	p-value
Not familiar with Kwak'wala	43.60	8.12e-05 *
Familiar with Kwak'wala	−17.86	0.228

Table 5.3: Per-page transcription time estimates from the linear mixed model comparing transcribers that had prior familiarity with Kwak'wala with those that did not. The time estimate is reduced by 17.86 minutes for a page when the user is familiar with Kwak'wala, indicating that target knowledge language might be useful to have in image transcription tasks. The p-value is > 0.05 for the estimate, which indicates that it is not statistically significant, likely because we only had two users that were familiar with the language.

manual transcription time. We also evaluate whether using the post-correction model is useful or just using an off-the-shelf tool like Ocular is sufficiently useful for transcribers. The LME model estimates for this comparison are in Table 5.2. We see that using our post-correction method in the transcription pipeline reduces manual correction time by 21%, indicating its significant utility to the downstream task of manually correcting the text.

Does prior familiarity with Kwak'wala and the Boas script affect transcription time? Beyond our primary analysis of the effect of using OCR, we also try to evaluate the extent to which the user's knowledge of the Kwak'wala language affects the speed of transcription. Table 5.3 demonstrates this comparison with results across all three task setups. The estimates show that this factor does play a role with the LME model estimate with a 40% reduction in transcription time for the group familiar with Kwak'wala. However, the p-value of this estimate is > 0.05, indicating that the result is not statistically significant – this is likely because only two transcribers in the user study had prior knowledge of the language and more data is needed to draw a statistically significant conclusion.

5.2.7 Subjective Feedback

After participants completed the transcription tasks, we asked them to fill out a short survey to describe their experience with the task. Note that, to avoid any bias, the participants were not told which OCR setup (off-the-shelf or post-correction) was used for each task. Therefore, the survey focused on understanding whether users observed any differences between typing from scratch or correcting transcriptions, but the questions did not distinguish between the two OCR-based setups. The full list of questions contained in the survey is in Section A.2.

We asked which of the setups led to faster completion of the tasks, and 100% of the participants perceived that correcting an OCR output was faster than typing the transcription from scratch. Some participants also provided feedback:

> *"Correcting is faster, as you only need to check for correctness and there is much less typing involved which requires most of the time"*
>
> *(user7, from Upwork, not familiar with Kwak'wala)*

> *"Correcting felt far more efficient!"*
>
> *(user2, linguistic researcher, familiar with Kwak'wala)*

However, even though it was slower, two out of the nine participants preferred typing out the text without the aid of an OCR output:

> *"I preferred typing the text from scratch, as searching for any editable text is a bit difficult. You need more effort for editing than writing."*
>
> *(user8, from Upwork, not familiar with Kwak'wala)*

However, the remaining seven transcribers provided strong feedback that correcting OCR outputs was the preferable task setup, for a variety of reasons:

> *"I vastly preferred correcting OCR outputs. It was so much faster, and also required less of an investment of attention."*
>
> *(user2, linguistic researcher, familiar with Kwak'wala)*

> *"I preferred correcting text - it's much faster. I can spend more mental energy making sure the characters are correct rather than wasting time on transcribing trivially-easy letters."*
>
> *(user5, computer science student, not familiar with Kwak'wala)*

> *"I prefer correcting text because typing from scratch is somehow tricky to follow line by line."*
>
> *(user9, from Upwork, not familiar with Kwak'wala)*

Overall, transcribers participating in the user study identified a reduction in time spent when the OCR outputs were utilized and the majority preferred the task setup not only because of the speed improvement but also because the OCR outputs allowed them to zoom in and fix specific errors rather than spending time on the entire image.

Additionally, we asked participants if any tasks seemed to be easier or more difficult than others. While several described correction as easier than typing from scratch, some transcribers focused on interesting language-specific and document-specific challenges:

> *"Correcting the predictions was easier."*
>
> *(user3, heritage language learner, familiar with Kwak'wala)*

> *"A few alphabets were difficult to annotate from the images. For example, it was difficult to differentiate between l and ł."*
>
> *(user6, from Upwork, not familiar with Kwak'wala)*

> *"image text was with small fonts."*
>
> *(user4, computer science student, not familar with Kwak'wala)*

> *"the hardest thing for me was identifying a particular character (ł) that is very faint in the original PDF. It is often difficult to tell if a character is ł or l. Because I have some knowledge of the language, I relied on that background knowledge at times, but this slowed down the correction process."*
>
> *(user2, linguistic researcher, familiar with Kwak'wala)*

In giving feedback about the keyboard practice tasks, all participants indicated that the practice task helped them learn the Hunt/Boas orthography and the keystroke mappings. Moreover, 100% of the participants stated that as they completed more tasks, they became faster at transcription. One participant *(user7, from Upwork, not familiar with Kwak'wala)* stated *"After transcribing a few pages, I became faster at typing with the keyboard and noticing the different accents and letters."* While the ordering of the tasks was not taken into account in our LME model because of the small amount of data in the current user study, we hope to understand the effect of task order on transcription time in future, larger-scale research.

5.3 Tools and Resources Created

- **OCR pipeline for the Boas orthography**: By using existing OCR tools such as Ocular (Berg-Kirkpatrick et al., 2013) and our post-correction model, we created an OCR pipeline for digitizing Kwak'wala text in the Hunt/Boas orthography with low transcription error rates, even when minimal amounts of training data are available.
- **Machine-readable text resources for Kwak'wala**: With our OCR pipeline, we have currently extracted Kwak'wala text from 1,500 pages from two volumes of Boas and Hunt (1921) that were previously only accessible as scanned images. The volumes also contain English text (each page has Kwak'wala text and its corresponding translation in English – see Figure 5.6 for an example), which we extracted using the Google Vision OCR, known to have strong performance on English documents (Fujii et al., 2017).

  Additionally, we automatically transliterate all the extracted Kwak'wala text, converting it from the Boas orthography to the modern, community-preferred U'mista orthography. We use `convertextract`[7] for this process, a tool that maps the Boas character sequences to their U'mista equivalent. While the transliteration is not perfect, because the outputs from our model are not completely error-free, Kwak'wala researchers and learners subjectively indicated that the transliterated text we generated is readable to a large extent.

  We release all 1,500 pages as a resource for Kwak'wala, with each page consisting of (1) machine-readable Kwak'wala text in the original Boas orthography; (2) machine-readable English translation of the Kwak'wala; (3) Kwak'wala text in the page transliterated to the U'mista orthography. With this release, researchers, language learners, and teachers in the Kwak'wala community can search through the extensive cultural information contained in the Hunt/Boas publications in both Kwak'wala and English. Additionally, the texts can be read by a much larger audience with the transliteration into a community-preferred orthography.
- **Virtual keyboard**: Although initially created to increase efficiency in the user study transcription task, the keyboard we developed for the Hunt/Boas orthography has proven valuable beyond the boundaries of this narrow use case. The keyboard package is now employed by multiple user groups beyond this project, including community-based teachers and learners of Kwak'wala and Bak'wa̲mk'ala; community and academic researchers

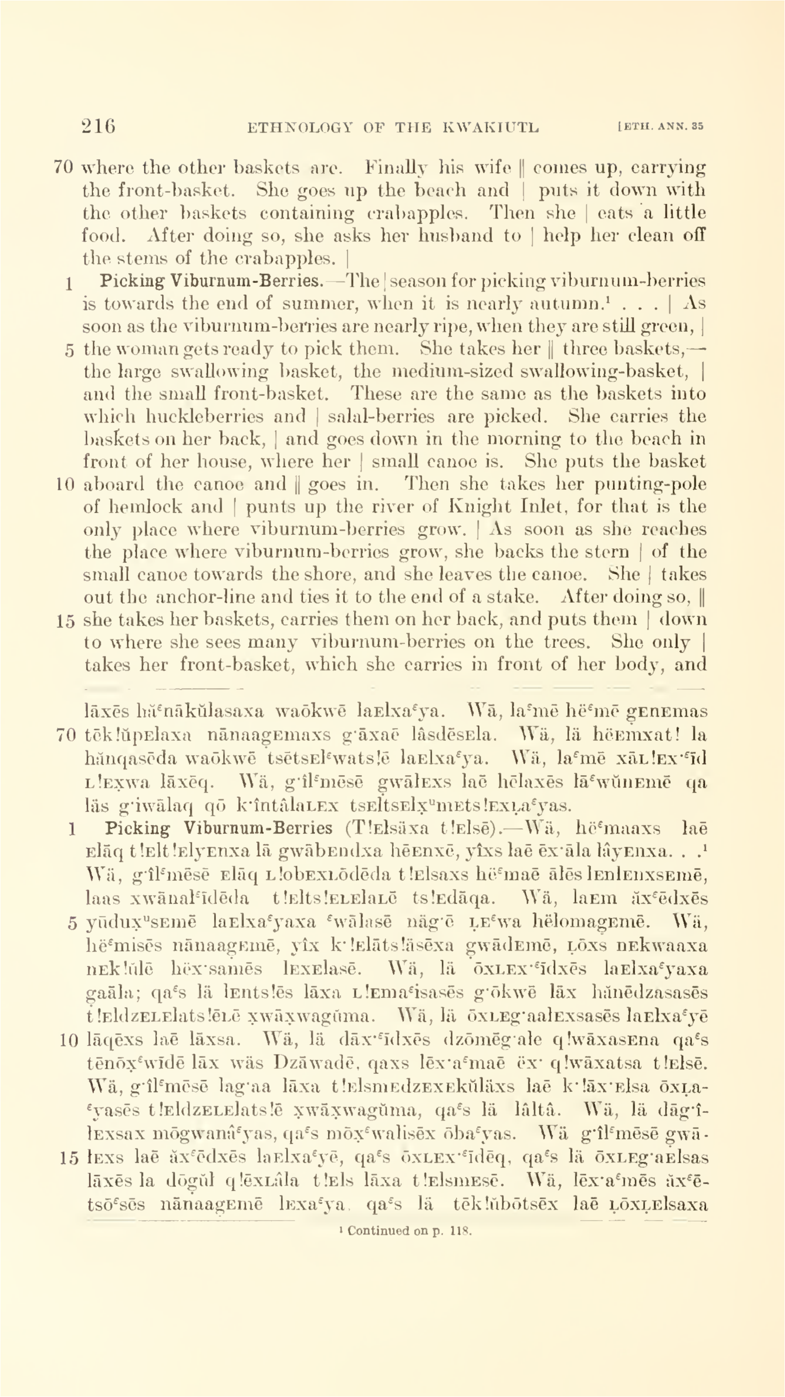
[7] https://github.com/roedoejet/convertextract

working with materials created by Boas and Hunt; and archivists and managers of repositories where these materials are held, including at the American Philosophical Society's Library.[8]

5.4 Summary

In previous chapters, we proposed post-correction models to improve OCR transcriptions on endangered language texts and saw a considerable reduction in error rates. In this chapter, we evaluate the utility of the models in a user-centric manner. We conduct a case study on the Kwak'wala language and focus on digitizing documents that use the complex Boas orthography. With a user study, we show the downstream utility of our proposed models in significantly reducing the time needed for manual transcription tasks. We also extract text from two published volumes of Kwak'wala linguistic and cultural documentation, making these important resources much more accessible to the community. While we focus on a single language in this case study, our results demonstrate the immense potential impact that improved OCR technologies can have on endangered language documentation and revitalization efforts.

[8] https://www.amphilsoc.org/library/CNAIR



216 ETHNOLOGY OF THE KWAKIUTL [ETH. ANN. 35

70 where the other baskets are. Finally his wife ‖ comes up, carrying
the front-basket. She goes up the beach and | puts it down with
the other baskets containing crabapples. Then she | eats a little
food. After doing so, she asks her husband to | help her clean off
the stems of the crabapples. |

1 **Picking Viburnum-Berries.**—The | season for picking viburnum-berries
is towards the end of summer, when it is nearly autumn.[1] . . . | As
soon as the viburnum-berries are nearly ripe, when they are still green, |
5 the woman gets ready to pick them. She takes her ‖ three baskets,—
the large swallowing basket, the medium-sized swallowing-basket, |
and the small front-basket. These are the same as the baskets into
which huckleberries and | salal-berries are picked. She carries the
baskets on her back, | and goes down in the morning to the beach in
front of her house, where her | small canoe is. She puts the basket
10 aboard the canoe and ‖ goes in. Then she takes her punting-pole
of hemlock and | punts up the river of Knight Inlet, for that is the
only place where viburnum-berries grow. | As soon as she reaches
the place where viburnum-berries grow, she backs the stern | of the
small canoe towards the shore, and she leaves the canoe. She | takes
out the anchor-line and ties it to the end of a stake. After doing so, ‖
15 she takes her baskets, carries them on her back, and puts them | down
to where she sees many viburnum-berries on the trees. She only |
takes her front-basket, which she carries in front of her body, and

lāxēs hăᵋnākŭlasaxa waōkwē laᴇlxaᵋya. Wä, laᵋmē hēᵋmē gᴇɴᴇmas
70 tēk!ŭpᴇlaxa nānaagᴇmaxs g·āxaē lâsdēsᴇla. Wä, lä hēᴇmxat! la
hănqasēda waōkwē tsētsᴇlᵋwats!ē laᴇlxaᵋya. Wä, laᵋmē xāʟ!ᴇx·ᵋīd
ʟ!ᴇx̣wa lāxēq. Wä, g·îlᵋmēsē gwāłᴇxs laē hēlaxēs łāᵋwŭɴᴇmē qa
läs g·iwālaq qō k·întâlaʟᴇx tsᴇltsᴇlx̣ᵘmᴇts!ᴇxʟ̣aᵋyas.

1 **Picking Viburnum-Berries** (T!ᴇlsäxa t!ᴇlsē).—Wä, hēᵋmaaxs łaē
ᴇlāq t!ᴇlt!ᴇlyᴇɴxa lā gwābᴇɴdxa hēᴇɴxē, yîxs łaē ēx·āla łâyᴇɴxa. . .[1]
Wä, g·îlᵋmēsē ᴇlāq ʟ!obᴇxʟōdēda t!ᴇlsaxs hēᵋmaē ālēs lᴇɴlᴇɴxsᴇmē,
laas xwānalᵋīdēda t!ᴇlts!ᴇʟᴇlaʟē ts!ᴇdāqa. Wä, laᴇm ăxᵋēdxēs
5 yūdux̣ᵘsᴇmē laᴇlxaᵋyaxa ᵋwālasē näg·ē ʟ̣ᴇᵋwa hēlomagᴇmē. Wä,
hēᵋmisēs nānaagᴇmē, yîx k·!ᴇlāts!äsēxa gwādᴇmē, ʟ̣ōxs ɴᴇkwaaxa
ɴᴇk!ŭlē hēx·samēs lᴇxᴇlasē. Wä, lä ōxʟᴇx·ᵋīdxēs laᴇlxaᵋyaxa
gaāla; qaᵋs lä lᴇɴts!ēs lāxa ʟ!ᴇmaᵋisasēs g·ōkwē lāx hănēdzasasēs
t!ᴇldzᴇʟᴇlats!ēʟē x̣wāx̣wagŭma. Wä, lä ōxʟᴇg·aałᴇxsasēs laᴇlxaᵋyē
10 lāqēxs laē lāxsa. Wä, lä dāx·ᵋīdxēs dzōmēg·ałe q!wāxasᴇɴa qaᵋs
tēnōx̣ᵋwīdē lāx wäs Dzāwadē, qaxs lēx·aᵋmaē ēx· q!wāxatsa t!ᴇlsē.
Wä, g·îlᵋmēsē lag·aa lāxa t!ᴇlsmᴇdzᴇxᴇkŭläxs laē k·!āx·ᴇlsa ōxʟ̣a-
ᵋyasēs t!ᴇldzᴇʟᴇlats!ē x̣wāx̣wagŭma, qaᵋs lä lâltâ. Wä, lä dāg·î-
łᴇxsax mōgwanâᵋyas, qaᵋs mōx̣ᵋwalisēx ōbaᵋyas. Wä g·îlᵋmēsē gwā-
15 łᴇxs laē ăxᵋēdxēs laᴇlxaᵋyē, qaᵋs ōxʟᴇx·ᵋīdēq, qaᵋs lä ōxʟᴇg·aᴇlsas
lāxēs la dōgŭł q!ēxʟâla t!ᴇls lāxa t!ᴇlsmᴇsē. Wä, lēx·aᵋmēs ăxᵋē-
tsōᵋsēs nānaagᴇmē lᴇxaᵋya, qaᵋs lä tēk!ŭbōtsēx laē ʟ̣ōxʟ̣ᴇlsaxa

[1] Continued on p. 118.

Figure 5.6: A page from the Hunt/Boas publications documenting the community's method for picking viburnum berries. As seen, the documents have bilingual text in English (top) and Kwak'wala (bottom), which are approximately aligned at the page-level. We convert both the English and the Kwak'wala text from 1,500 pages into a machine-readable format, enabling search and retrieval of information in both languages.

Chapter 6

Making OCR and Post-Correction Models More Accessible

In previous chapters, we build OCR post-correction models to improve OCR accuracy in very low-resource settings as well as evaluate the utility of the models with an extensive case study on Kwak'wala, an endangered language spoken in North America. We demonstrate that having technology that produces highly accurate OCR transcriptions can be very beneficial to communities that speak endangered languages, particularly for language education, documentation, and revitalization efforts.

Many OCR and post-correction technologies, including the models we develop in previous chapters, are publicly available as open-source software, and are typically accessible through the command line (e.g., Python packages) or code modules that query APIs. Even popular tools like Google Vision OCR and Tesseract require familiarity with programming, installing dependencies, and the command line to apply available off-the-shelf models. Moreover, training a new model (e.g., with our post-correction software that is currently available only as Python modules) typically involves several steps and requires substantial technical expertise.

While researchers and practitioners with the necessary technical proficiency and programming skill can use these systems for training and inference, much of the target audience for the models and OCR pipelines developed in this thesis is outside technical fields. If our software is only available as command line scripts and code modules, it limits accessibility to linguists, language learners, teachers, archivists, and community-based researchers who could greatly benefit from improved OCR for endangered languages, but may not be familiar with programming tools and packages.

In this chapter, to enable easier access to OCR and post-correction systems, we build a web interface that allows the use of these tools without the need for writing or understanding any code. The interface is designed to make it straightforward to apply these technologies on documents that contain endangered language text, with functionality that enables:

- Inference with off-the-shelf OCR, such as Google Vision and Tesseract, with which users can obtain transcriptions for scanned images – either as a single step or as a first-pass for subsequent post-correction.
- Training and inference with the post-correction models proposed in Chapter 3 and Chapter 4, for users to improve existing OCR transcriptions.

Web interfaces exist for some off-the-shelf tools, like the EasyOCR demo[1] and Google Vision demo.[2] However, these are limited in their functionality in that they only allow inference on a single image at a time and they only support one OCR system. The interface we present in this chapter is, to the best of our knowledge, the first to support no-code inference with several state-of-the-art OCR and post-correction methods as well as no-code training for post-correction models. The interface, thus, allows users to compare and contrast results across systems and decide which work best for their target documents, orthographies, and languages.

In the sections that follow, we describe the interface (which is implemented as a web application), details of the various functionalities that it supports as well as workflows of how a potential user might interact with the application based on the requirements for their specific use case as workflows might vary depending on the target document or language. We also release the interface as a public web application that can be accessed by anyone with an internet connection, in the hope of enabling use of these technologies by a much larger set of people.

6.1 Functionalities

Prediction with off-the-shelf OCR The interface supports inference with existing off-the-shelf OCR tools that are typically available only through a command line interface or a code package. This functionality takes scanned documents, either in image formats or a PDF, and returns the prediction from the selected OCR system on each image or page in the PDF.

No-code access to these existing tools is useful for practitioners because several of these systems support a large number of languages (typically 80–100) and many scripts. This makes them applicable to a broad range of documents because the target language is not directly

[1] https://www.jaided.ai/easyocr/

[2] https://cloud.google.com/vision/docs/drag-and-drop

supported, the OCR might still be reasonably accurate if the script is known to the model. The interface, in its current form, only enables prediction on input documents with these off-the-shelf models and does not allow training or fine-tuning (which some packages like Tesseract include functionality for).

To illustrate the utility of this functionality, consider the status quo, where if a user had a large number of images that they wanted to apply the Google Vision OCR on, they would have to either write code to use the API, which requires technical proficiency or upload images one at a time to the online demo,[3] which is time-consuming for large documents. With our interface, the user can simply upload all target images together and get the OCR predictions in a single step. Additionally, the user would be able to experiment with different off-the-shelf systems within the same interface to see which one works best for their documents (e.g., comparing predictions from Google Vision with those from Tesseract).

Training OCR post-correction models The interface also supports the training of post-correction models based on the methods presented earlier in this thesis. The user can train either a supervised model discussed in Chapter 3 or a semi-supervised model discussed in Chapter 4 for their target language or orthography, with any existing first-pass OCR they might already have. This functionality requires first-pass OCR and corresponding manually transcribed text for supervised training, along with first-pass OCR from any available unlabeled pages for pretraining and semi-supervised learning, and returns a trained post-correction model. Technical aspects of the training, such as preprocessing, data splits, and hyperparameter tuning, are handled in the backend of the interface and users only need to focus on providing a minimal amount of training data.

Before building this interface, training post-correction models with our proposed method involved downloading software modules built in Python, installing dependencies, as well as understanding and modifying the scripts to run the modules. With the interface, the user does not need these technical skills and can train new models directly through the web application. Teachers, speakers, and researchers of endangered languages can thus easily train a model to improve OCR performance beyond off-the-shelf tools as our proposed methods improve accuracy even in very low-resource settings.

Prediction with trained post-correction models With this functionality, trained models can be used for making predictions on new data – the input here would be the first-pass OCR

[3] https://cloud.google.com/vision/docs/drag-and-drop

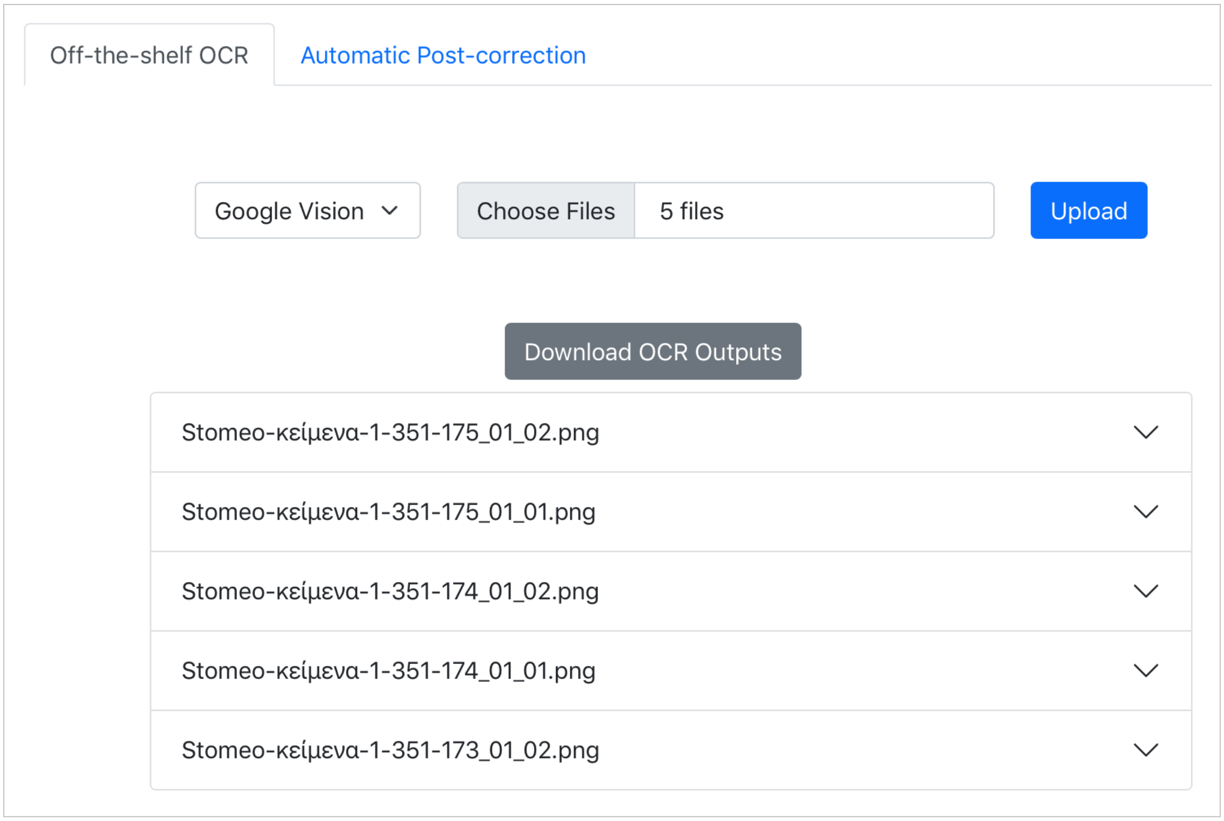
for the set of pages that need to be transcribed and the output would be the post-corrected text. Having this inference functionality in the interface makes it easier for users to apply trained post-correction models to large collections of documents and improve the accuracy of text transcriptions (and consequently bettering readability and downstream usability of the documents) without the need for programming or technical proficiency.

Workflows The OCR interface can be used in several different ways depending on the specifics of the target use case because the functionalities operate independently. The workflow which leverages all the functionalities of the interface starts with the user uploading the input scanned document(s) to the web application and obtaining the first-pass OCR transcription for each page from an off-the-shelf system. Next, the user uploads training data for the OCR post-correction model, which consists of the first-pass OCR and parallel manual transcriptions. The user can also upload first-pass text for pages that are not manually transcribed (unlabeled data), and these can be used for semi-supervised learning. This process results in a trained model as output. Finally, the trained model can be used for inference, where the user uploads first-pass OCR for all the pages and documents that need to be corrected.

However, the user may not need to use all the available features of the interface. For example, a possible workflow for documents on which off-the-shelf models have high accuracy likely does not need to include the post-correction training and inference steps (e.g., using the Google Vision system typically results in very low error rates for printed Latin script text). The user may also want to try out the various off-the-shelf models supported in the interface to determine the best one for the target data. Similarly, if the user already has OCR transcriptions for the documents that are not very accurate (as many libraries and archives often do) and wants to improve them with post-correction, getting a first-pass OCR from the interface is no longer necessary. The independence of the components makes the interface adaptable to different workflows and use cases, and allows the user to easily experiment with various settings and OCR pipelines.

6.2 Implementation

The interface is implemented as a simple web application, making it straightforward to access by anyone with an internet connection. Like most web applications, the implementation includes *frontend* and *backend* components. The user interacts with the frontend through a web browser, and the backend executes commands received from the frontend based on what func-



Google Vision

5 files

Stomeo-κείμενα-1-351-175_01_02.png

Stomeo-κείμενα-1-351-175_01_01.png

Stomeo-κείμενα-1-351-174_01_02.png

Stomeo-κείμενα-1-351-174_01_01.png

Stomeo-κείμενα-1-351-173_01_02.png

Figure 6.1: The interface frontend for inference with off-the-shelf OCR systems. The user can choose the OCR system as well as upload images or PDF files to transcribe. When inference is complete, the user can download the outputs as text files. Note the tabbed interface that allows users to experiment with off-the-shelf tools and our post-correction models independently.

tionalities the user chooses to use. This abstracts the technical details of the OCR models away from the end user and ensures the user only needs to focus on components specific to their target data (e.g., uploading images and exporting OCR outputs).

6.2.1 Frontend

We develop a frontend for the web application which is the public-facing web page that users view and interact with. The frontend is simply a graphical interface for users to communicate with the backend and execute their desired workflow, by uploading and downloading data as well as selecting training and inference functions with various models. The frontend is written in Javascript using the React-Bootstrap framework,[4] which contains implementations of several

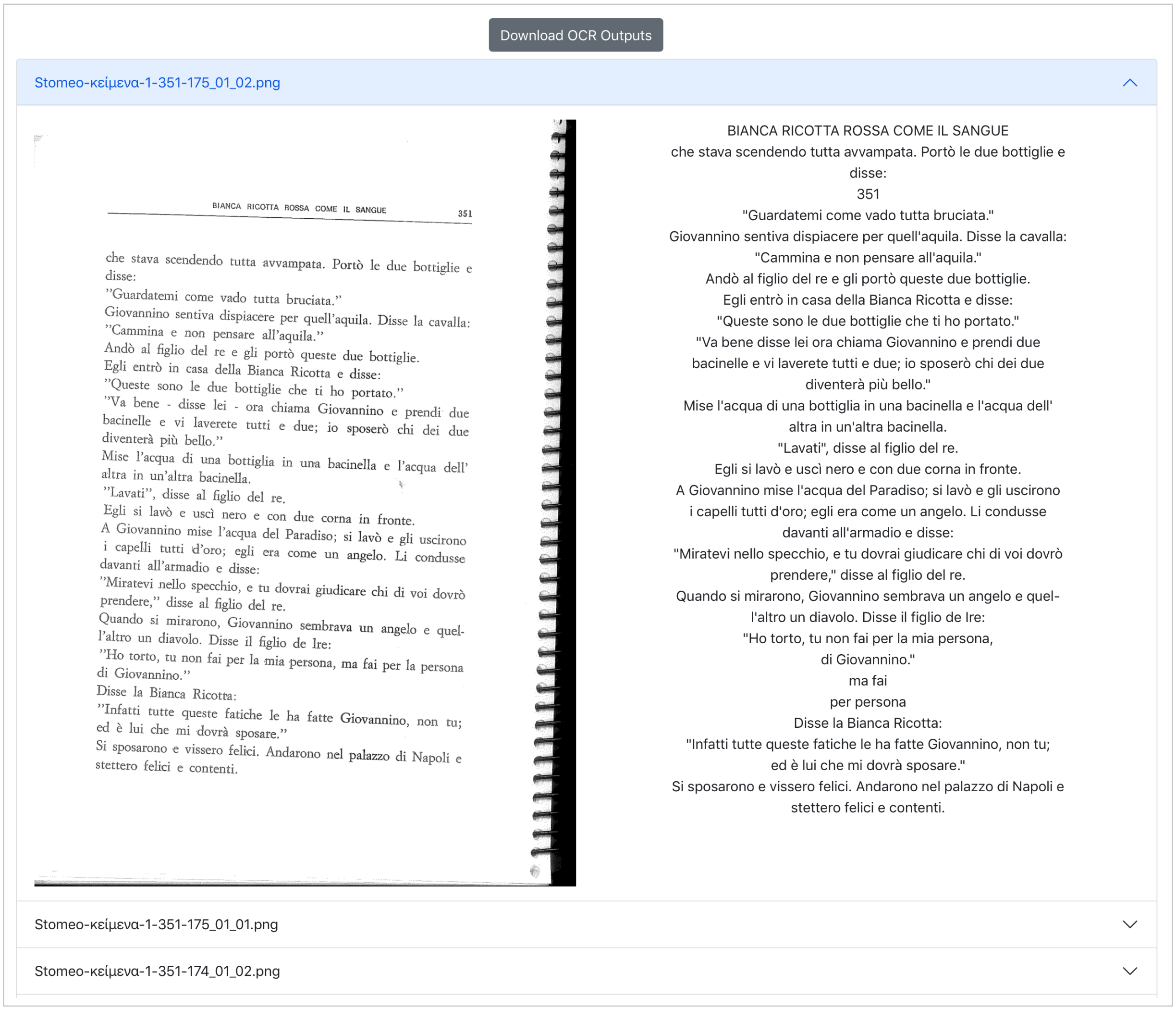
[4] https://react-bootstrap.github.io

useful UI components in its library. The axios package[5] is used for communication of data and instructions between the frontend and backend. The frontend is designed to reflect the flexibility of the workflow, where the off-the-shelf OCR tools and the post-correction functionalities can be used independently, all within the same interface.

Off-the-shelf OCR A screenshot of the frontend for the inference functionality with an off-the-shelf OCR system is shown in Figure 6.1. As seen, the interface is simple – first, the user selects the OCR system they want to use (the figure shows "Google Vision" selected). Next, the user selects the files they need digitized (multiple files can be selected in a single upload and PDFs, as well as image formats such as PNG and JPG, are permitted) and clicks the "Upload" button that sends the information to the backend. The backend server then executes commands to preprocess the data, if necessary, and apply the selected OCR model on the input data. Finally, the outputs from the system are returned to the frontend and the user is also able to export all the transcriptions in text files, one per page, to their local computer storage by clicking the "Download OCR Outputs" button. Additionally, the outputs are displayed on the web page in an easy-to-read format that aligns each image with its corresponding transcription using a vertical stack of expandable headers (one for each input file), as seen in Figure 6.2. This enables easy viewing of the outputs within the interface and users can inspect them without having to download any files to local storage.

OCR Post-correction The frontend for training and inference with the post-correction model is similar, as seen in Figure 6.3. If the user wants to train a model, they upload manually annotated data in text files ("Training Data") and, optionally, unlabeled data for semi-supervised learning ("Unlabeled Data") and click the "Train new model" button to initiate training in the backend. The user also provides an email address, at which a notification will be sent when training is complete and the trained model file can be downloaded. For the inference functionality, the user selects the trained model to be used ("Model File") as well as the first-pass OCR files to be corrected, both of which are uploaded to the backend when the user clicks the "Apply model" button. Similar to the training functionality, an email notification is sent to the user after prediction on all input files is complete.

[5] https://www.npmjs.com/package/axios



BIANCA RICOTTA ROSSA COME IL SANGUE 351

che stava scendendo tutta avvampata. Portò le due bottiglie e disse:
”Guardatemi come vado tutta bruciata.”
Giovannino sentiva dispiacere per quell’aquila. Disse la cavalla:
”Cammina e non pensare all’aquila.”
Andò al figlio del re e gli portò queste due bottiglie.
Egli entrò in casa della Bianca Ricotta e disse:
”Queste sono le due bottiglie che ti ho portato.”
”Va bene - disse lei - ora chiama Giovannino e prendi due bacinelle e vi laverete tutti e due; io sposerò chi dei due diventerà più bello.”
Mise l’acqua di una bottiglia in una bacinella e l’acqua dell’ altra in un’altra bacinella.
”Lavati”, disse al figlio del re.
Egli si lavò e uscì nero e con due corna in fronte.
A Giovannino mise l’acqua del Paradiso; si lavò e gli uscirono i capelli tutti d’oro; egli era come un angelo. Li condusse davanti all’armadio e disse:
”Miratevi nello specchio, e tu dovrai giudicare chi di voi dovrò prendere,” disse al figlio del re.
Quando si mirarono, Giovannino sembrava un angelo e quell’altro un diavolo. Disse il figlio de Ire:
”Ho torto, tu non fai per la mia persona, ma fai per la persona di Giovannino.”
Disse la Bianca Ricotta:
”Infatti tutte queste fatiche le ha fatte Giovannino, non tu; ed è lui che mi dovrà sposare.”
Si sposarono e vissero felici. Andarono nel palazzo di Napoli e stettero felici e contenti.

```
BIANCA RICOTTA ROSSA COME IL SANGUE
che stava scendendo tutta avvampata. Portò le due bottiglie e
disse:
351
"Guardatemi come vado tutta bruciata."
Giovannino sentiva dispiacere per quell'aquila. Disse la cavalla:
"Cammina e non pensare all'aquila."
Andò al figlio del re e gli portò queste due bottiglie.
Egli entrò in casa della Bianca Ricotta e disse:
"Queste sono le due bottiglie che ti ho portato."
"Va bene disse lei ora chiama Giovannino e prendi due
bacinelle e vi laverete tutti e due; io sposerò chi dei due
diventerà più bello."
Mise l'acqua di una bottiglia in una bacinella e l'acqua dell'
altra in un'altra bacinella.
"Lavati", disse al figlio del re.
Egli si lavò e uscì nero e con due corna in fronte.
A Giovannino mise l'acqua del Paradiso; si lavò e gli uscirono
i capelli tutti d'oro; egli era come un angelo. Li condusse
davanti all'armadio e disse:
"Miratevi nello specchio, e tu dovrai giudicare chi di voi dovrò
prendere," disse al figlio del re.
Quando si mirarono, Giovannino sembrava un angelo e quel-
l'altro un diavolo. Disse il figlio de Ire:
"Ho torto, tu non fai per la mia persona,
di Giovannino."
ma fai
per persona
Disse la Bianca Ricotta:
"Infatti tutte queste fatiche le ha fatte Giovannino, non tu;
ed è lui che mi dovrà sposare."
Si sposarono e vissero felici. Andarono nel palazzo di Napoli e
stettero felici e contenti.
```

Stomeo-κείμενα-1-351-175_01_01.png
Stomeo-κείμενα-1-351-174_01_02.png

Figure 6.2: The interface frontend displays outputs from off-the-shelf OCR systems in a vertical stack of expandable web containers. Each container has one of the input images and its corresponding transcription, and users can interact with them to expand and collapse outputs.

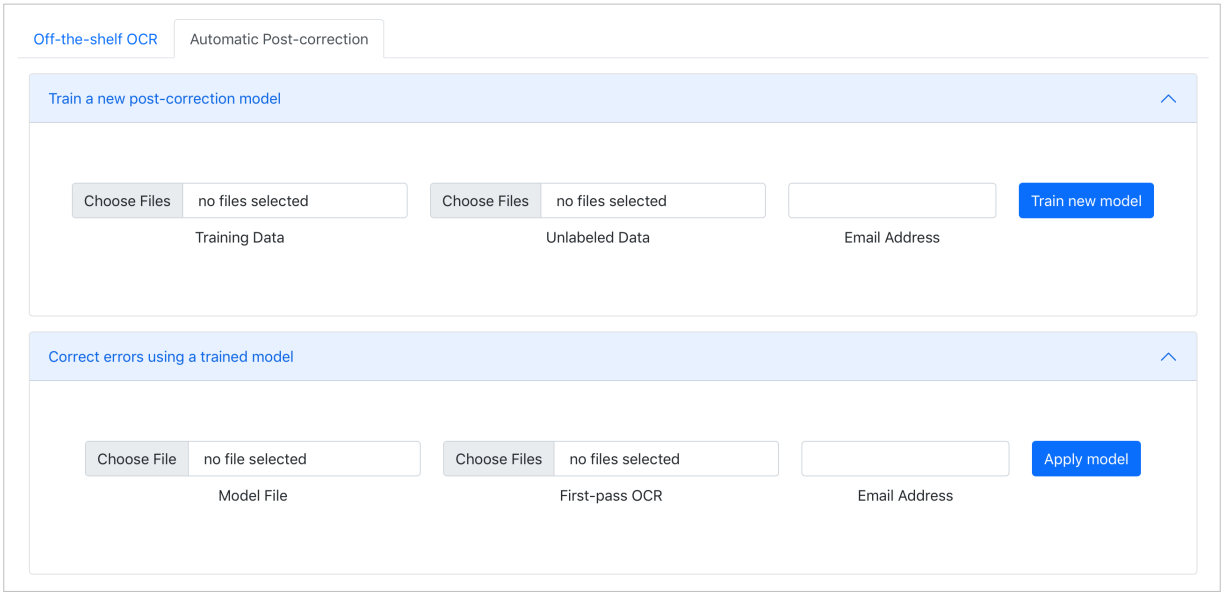


Figure 6.3: The post-correction tab in the interface is the frontend for training and prediction with the post-correction methods we propose in this thesis. It allows users to upload files for training a new model as well as for making predictions to correct first-pass OCR files with an already trained model. The user's email address is used to send notification when training and inference jobs are complete.

6.2.2 Backend

We use an existing backend framework, the CMU Linguistic Annotation Backend (CMULAB),[6] for hosting code and scripts for the OCR technologies as well as executing the functions supported by our interface. CMULAB is implemented in Django,[7] a server-side web framework written in Python.

6.3 Using the Interface

The interface is designed to be easy to use without any technical or programming proficiency. In this section, we detail how the interface can be used to train OCR post-correction models on new datasets and languages with a pipeline covering all functionalities that are supported in the interface.

A user of the interface likely has a scanned document or set of documents that text needs to be extracted from. The first step in building a post-correction model involves obtaining a first-pass OCR on these documents. The interface's functionality that supports off-the-shelf OCR models can be used to get a first-pass transcription, following these steps:

- The user needs to ensure the input documents are in a format accepted by the interface: either in PDF or image format.
- Optionally, the user can preprocess the scanned images before applying an off-the-shelf OCR tool. Layout analysis of the documents, such as cropping or slicing the image as well as image enhancement techniques like binarization and improving contrast, can potentially lead to better OCR outputs. For documents that contain translations, cropping is often useful in getting separate first-pass transcriptions for each languages. The interface does not support preprocessing – the user can use visual layout analysis tools such as LAREX[8] for semi-automatic processing of the target documents.
- The scanned images are then to be uploaded in the interface, where the off-the-shelf OCR system selected by the user is applied to get transcriptions (see Figure 6.1).

The interface enables downloading the OCR outputs in text files. Once these are obtained, the next step is constructing a dataset to train the OCR post-correction model, since our proposed methods require manually annotated pages for training. Since the model is designed for

[6] https://github.com/neulab/cmulab

[7] https://www.djangoproject.com/

[8] https://github.com/OCR4all/LAREX

a low-resource setting, a small number of manually annotated pages (≈10 pages) is typically sufficient to train a model, although more annotations will likely lead to a better-performing model. The steps for creating the dataset are:

- The user selects the subset of pages in the documents that will be manually corrected. The remaining uncorrected pages will be used for semi-supervised learning (as described in Chapter 4).
- The selected pages are then manually transcribed – thus, the "labeled" training dataset has the first-pass OCR (the *source*) and the corresponding manually corrected transcription of the document (the *target*). The model training typically works best if these corresponding texts are aligned at the sentence-level or at the line-level. If the document contains translations and the user would like to use the multi-source setup described in Chapter 3, the first-pass of the translated text also needs to be aligned in the dataset. These texts form the "Training Data" input for training the post-correction model, seen in Figure 6.3.
- For the uncorrected pages, the first-pass OCR for both the target language text and the translated text (if a multi-source model is needed) forms the "Unlabeled Data" input seen in Figure 6.3. These pages are used for pretraining and semi-supervised learning of the model. Using unlabeled data is optional, but highly recommended – as seen in Chapter 3 and Chapter 4, pretraining and semi-supervised learning techniques improve performance significantly in low-resource settings.
- The input datasets are then uploaded on the interface, along with the user's email address for notification when the training operation is complete. Log files generated by the backend training process inform the user about character and word error rates of the trained model on an evaluation set randomly selected from the labeled data.

With a trained post-correction model, the user is able to apply it to improve OCR performance on new documents. In order to do this, the first step involves obtaining a first-pass OCR and processing the transcriptions into sentence-level or line-level text files as described above. For a multi-source model, the first-pass of the translated text aligned with the target language text needs to be included in the input data. As seen in Figure 6.3, the first-pass is then uploaded to the interface, along with the trained model, and the backend launches a script to perform prediction on the inputs. A link to the corrected transcriptions is emailed to the user after prediction on all the input files is complete.

6.4 Future Directions for Advanced Functionality

In this chapter, we develop a web application for users to access the functionalities of state-of-the-art OCR and post-correction models without needing to write or understand any code. Particularly for communities and researchers of endangered languages, obtaining machine-readable transcriptions of historical documentation and language learning material can support revitalization and preservation efforts as well as help build downstream language technologies. The interface we present here can have a far-reaching impact by reducing the time, effort, and technical knowledge needed to apply existing OCR models and systems to new languages and writing systems.

While the web application described in previous sections is a research prototype, the interface is designed to be extendable to include new features and functionalities that would be useful to end users. In future development, we envision that the interface will incorporate advanced functions that account for challenging text recognition scenarios as well as improved user-friendliness. This could include features such as:

- better data and project management, where users have a personalized interface and storage for their data and models, which will make access more user-friendly than the current paradigm of uploading files from local storage at each step;
- automatically leveraging publicly available data sources, where the application identifies and uses any text data already available on the web in the target language or closely-related languages to train OCR post-correction models. Much prior work in post-correction uses existing machine-readable text with randomly generated noise to build synthetic training data for post-correction (Dong and Smith, 2018; Hakala et al., 2019). Additionally, Tjuatja et al. (2021) demonstrate some success in using target language data from the web and transfer learning from high-resource languages to improve post-correction models. Text data could be obtained from large multilingual corpus collections like PanLex, OPUS, and Wikipedia as well as linguistic archives. This functionality could be particularly useful for our target audience, as obtaining even a small number of manually transcribed images for low-resourced and endangered languages is often difficult;
- an annotation interface for active learning, for users to get initial outputs from a baseline OCR or post-correction system, manually correct some outputs (which could be selected using active learning), and re-train the post-correction model with the newly annotated data to improve performance, all within a single web interface;

- document and image analysis tools that allow users to crop images, apply image processing that can improve OCR performance like binarization and increasing contrast, and add bounding boxes to segment the image properly as OCR systems often return transcriptions in incorrect reading order when used on documents with complex layouts.

Chapter 7

Conclusion

State-of-the-art optical character recognition systems are highly performant and produce very accurate transcriptions in a large number of languages. Training such strong OCR models, however, requires substantial resources such as transcribed images and machine-readable text. Existing techniques are typically not designed to work well in low-resource scenarios, and this precludes building models for high-quality text extraction in languages that do not have sufficient training datasets.

In this thesis, we address the task of improving OCR transcriptions in very low-resource settings. We specifically focus on improving models for endangered languages, because there are thousands of printed books and documents that contain text in endangered languages, but most of these only exist as scanned images. Converting these documents to machine-readable text has a variety of benefits for language documentation and revitalization, including improving accessibility of the documents, enabling search within the documents, and providing text data for building NLP systems in endangered languages. The work presented in this thesis is, to the best of our knowledge, the first to introduce and address this task, with several research contributions that have had practical impact.

7.1 Summary of Contributions and Impact

Benchmark Dataset

This thesis presents the creation of a benchmark dataset for the task of OCR on endangered language documents in Chapter 2 containing manual transcriptions for printed books in a set of typologically-, orthographically-, and geographically-diverse endangered languages – Ainu, Griko, Kwak'wala, and Yakkha. We demonstrate that the performance of existing OCR systems is considerably lower on these languages than the high-resource languages prior research in OCR has focused on. With an in-depth analysis of outputs, we conclude that the disparity in performance occurs because of the lack of resources as well as some characteristics of endangered language writing systems like mixed scripts and uncommon diacritics. These results highlight the necessity for improving OCR technologies for endangered languages.

Methods for Low-Resource OCR Post-Correction

To improve the quality of OCR transcriptions, we use OCR post-correction: a text-based sequence-to-sequence technique that fixes errors in existing OCR outputs. Chapter 3 presents a supervised character-level model that includes a multi-source encoder and structural biases to enable better learning in low-resource settings. We show that with a very small amount of training data (a few hundred lines), the model obtains a relative reduction of 34.6% and 32.4% in character and word error rates respectively over existing OCR systems.

In Chapter 4, we improve the post-correction model using semi-supervised learning. The technique includes self-training the model by making predictions on raw images and using them as pseudo-training data for the model. We also propose "lexically-aware decoding" that creates a word list from the self-training predictions and uses frequency-based information to enforce word-level consistency in the model's outputs. This is done by combining probabilities from the LSTM decoder and a count-based language model represented with a WFSA. The addition of these components to the model results in a 23% average reduction in character error rate and a 17.5% average reduction in word error rate over the supervised method.

Our work on developing low-resource methods for OCR post-correction lends research insights beyond this specific task. First, we see that using additional sources of information, like translations, and adding biases to the model based on the structure of the task are ways to improve learning in very low-resource settings – these ideas can potentially be applied to developing models for other tasks as well, given that the resources needed for most state-of-the-art NLP systems are not available in endangered languages. Additionally, we find that designing techniques to use unlabeled data efficiently, such as our proposed lexically-aware decoding method, can improve performance without requiring any manual annotation. Finally, we see that our empirical evaluation indicates there is utility in combining neural methods with n-gram-based models. Techniques that combine different types of models could have an impact on other applications where there is not enough data to train a strong neural model.

User-Centric Evaluation

In Chapter 5, we address evaluation of OCR technologies for endangered languages with a user-centric approach. Specifically, we note that language documentation efforts often include manual transcription of scanned images, and we evaluate whether our proposed post-correction techniques can assist transcribers and reduce the amount of time spent on the task. In a case study on the Kwak'wala language, we see that using any form of OCR reduces the estimated time for transcription by over 33 minutes per page. Further, using the post-correction models developed in this thesis, the time estimate is reduced by nearly 7 minutes per page over off-the-shelf OCR tools. These results indicate the importance and downstream potential benefits of developing better OCR pipelines for endangered languages in supporting language revitalization and preservation programs by reducing the manual effort needed to produce machine-readable documents.

Tools and Resources

With the methods developed in this thesis, we have created multiple tools and resources that are released for public use, in the hope of furthering research and development for low-resource OCR and post-correction as well as enabling large-scale digitization of documents in many diverse endangered languages.

- **Dataset for the task** We are the first to introduce the task of OCR on endangered language documents, illustrating the challenges involved in using current state-of-the-art methods and public releasing a benchmark evaluation dataset for further research and improvement in performance on the task.
- **Post-correction software** We release Python modules to train post-correction models on new languages and datasets with the methods presented in this thesis. The software is released as a public repository on GitHub.[1] Beyond the languages in our dataset (i.e., Ainu, Griko, Kwak'wala, Yakkha), the software has been applied to many more low-resource and endangered languages by researchers around the globe: including Aghem; Babanki; Bhutia; Kom; Igbo; Oku; Piaroa; Pintupi-Luritja (Disbray et al., 2022); Quechua (at the *New Languages for NLP Workshop*);[2] Sanksrit; and Tibetan.
- **Kwak'wala language resources** We release multiple resources for the Kwak'wala

[1] https://github.com/shrutirij/ocr-post-correction

[2] https://newnlp.princeton.edu/

language as part of the case study in Chapter 5. These include (1) an OCR pipeline to produce high-quality transcriptions for Kwak'wala documents that use the Boas orthography; (2) extracted text for hundreds of pages from the Boas/Hunt publications that community-based researchers are currently using for a variety of purposes, including creating teaching curricula and retrieving cultural information; and (3) a keyboard for the Boas orthography that is being used by community members and archivists, including those at the American Philosophical Society's Library.[3]

- **Web application for using OCR technologies** In Chapter 6 we present the development of a web interface for users to easily use state-of-the-art OCR tools, including off-the-shelf systems and our proposed post-correction method. The interface is the first of its kind in that allows users to experiment with different OCR systems without any technical knowledge or coding proficiency, making them accessible to a much broader audience and, hopefully, allowing easy training of models and building OCR pipelines for more languages at scale.

7.2 Limitations and Future Directions

While the contributions in this thesis take a step toward improving OCR for endangered languages and using OCR technologies to make an impact in endangered language communities, there are multiple research directions as extensions of our work that can further advance the state-of-the-art in low-resource text recognition and have a broad practical impact on linguistic documentation, language revitalization programs, and NLP applications.

Larger Benchmark Datasets for Evaluation

The dataset we present in this thesis contains documents from four endangered languages. Future work could consider further development of this benchmark to a larger set of languages that covers diverse geographies and orthographies as well as different document layouts such as dictionaries and handwritten documents. Documents for expanding the dataset can be sourced from publicly available sources like the Internet Archive,[4] AILLA,[5] and ELAR[6] – all of which contain thousands of documents with endangered language text. An expanded benchmark

[3] https://www.amphilsoc.org/library/CNAIR

[4] http://archive.org/

[5] https://ailla.utexas.org/

[6] https://www.elararchive.org/

dataset would make it possible for researchers to do OCR and post-correction model evaluation that tests generalizability on a broad range of endangered languages and document types.

Reducing Reliance on Manual Transcription

The post-correction methods we develop use a small amount of manually transcribed data for training. While the required data size is orders of magnitude smaller than what is needed to train existing OCR models, it is challenging to obtain *any* manual transcription for some endangered languages. Reducing the reliance of the model on these manual transcriptions is an important future direction that will make expansion to new languages much easier.

The development of an active learning technique for OCR post-correction could be one way to reduce the amount of manual transcription needed for training. Starting with a model trained on very little data, active learning would select specific sentences or examples that are most useful for the model to learn from and only obtain manual annotation for those. These can then be added to the training set to update the model. This is a process that can be repeated iteratively, and the model can incrementally learn to correct its mistakes. The manual annotation will also likely be faster here because the transcriber can correct outputs from the previous model as opposed to transcribing from scratch.

Another avenue for improvement without manual transcription is effectively utilizing available linguistic resources. The majority of endangered languages have existing language documentation, which often includes materials like word lists, grammar, and morphological information (Boerger and Stutzman, 2018; Bird, 2020). An extension of our proposed lexically-aware decoding method could build a WFSA with word lists from the target language and use it as in Chapter 4 for joint decoding with the neural post-correction model, encouraging the model to generate words from the manually-crafted lists. Additionally, there is prior work on creating synthetic training data for OCR post-correction, by adding random noise to machine-readable text in the target language (Dong and Smith, 2018; Hakala et al., 2019). While large amounts of text data are usually not available for endangered languages, word lists could be used as a starting point for generating data with these methods to reduce the amount of manually annotated pages needed for training a strong model. The research challenges here include the lack of enough text to generate diverse enough synthetic data and the fact that the data will only contain single words, but we want the post-correction model to operate on full lines or sentences of text.

Other linguistic resources, such as morphological analyzers, could also be used to improve

OCR post-correction performance. For morphologically-rich languages like Kwak'wala, it is challenging to automatically or manually create a high-coverage lexicon for the lexically-aware decoding method. While the character-level unknown word model used in Chapter 4 does help improve prediction of unknown words, it is sub-optimal as several of the unknown words may be inflections of words already seen by the model. An extension of our proposed method could use "morphologically-aware decoding", with generation of post-corrected text that is constrained by a hand-crafted morphological analyzer (Silfverberg and Rueter, 2015; Matthews et al., 2018), if these resources exist in the target language. If hand-crafted analyzers are unavailable, morphological information can be incorporated by generating inflections of words in the unlabeled dataset for lexically-aware decoding.[7]

Reducing the reliance of the post-correction method on manually annotated data will make it easier for users to build models for new languages and datasets by lowering the amount of time and effort required to create a labeled dataset (as described in Section 6.3).

Advanced Functionality in the Web Interface

We present a prototype of a web interface for using OCR technologies in Chapter 6 but recognize that there are several useful functionalities that the application does not currently support. Future work could focus on further development of the application to serve users with all features required for an end-to-end OCR pipeline. Some of these are described in detail in Section 6.4, and include personalized data and project management for each user; automatic collection of publicly available data for the target language and related languages to improve training of the model; allowing users to manually transcribe images within the interface (we previously used Label Studio for transcription, as discussed in Chapter 5), which could also enable active learning and iterative training techniques; and document analysis tools like cropping and setting bounding boxes, which could be implemented by integrating an open-source layout analysis system like LAREX[8] into the interface. Preprocessing scanned images is often necessary to improve OCR performance, and incorporating these tools into the interface would streamline the training process and improve overall user experience with our proposed web application.

[7] Recent work for low-resource languages has shown that a very small number of manually tagged words is required for reasonable performance on morphological inflection generation (Anastasopoulos and Neubig, 2019).

[8] https://github.com/OCR4all/LAREX

Text Data for Downstream NLP Tasks

Extracting high-quality transcriptions from documents in a large number of endangered languages has the potential to greatly improve the representation of these languages in NLP technologies because the lack of machine-readable text has left these languages behind in state-of-the-art NLP models (Joshi et al., 2020). Text data can be used for self-supervised learning of language models, building predictive keyboards and spell-checking tools, and annotating training and evaluation sets for downstream NLP tasks. Communities that speak endangered languages are often invested in using language technologies to support education and revitalization programs, indicating the immense potential benefits of including these languages in modern NLP systems: for example, Kwak'wala language researchers intend to use the text extracted from the Hunt/Boas publications and other documents to train automatic speech recognition systems and reduce the effort needed to manually transcribe recorded Kwak'wala speech. Future research directions could include developing training strategies to best use OCR-extracted texts in improving NLP systems as well as understanding how effective the utilization of these texts is in overcoming the challenges of low-resource learning and whether this varies for different NLP tasks. It would also be useful to understand how tuning and evaluation of the OCR pipeline affects downstream tasks – for instance, a model tuned to optimize word error rate, as opposed to character error rate, might be preferred for search applications.

Appendix A

A.1 Keyboard for the Boas/Hunt Orthography

For the user study described in Section 5.2, we designed a keyboard for the Boas/Hunt orthography in order to make transcription more efficient.

The keyboard is developed using open-source software Keyman[1] and it maps characters in the Boas orthography to the user's computer keyboard. Keyman also provides an on-screen keyboard to see the mapped layout. The keyboard has been publicly released as a Keyman package.[2] We briefly describe the layout and usage of the keyboard below:

- Standard English keyboard alphabet and numbers remain in the same position (A-Z, a-z, 0-9) because the Boas orthography uses several Latin script characters.
- The special characters, diacritics, and digraphs of the Boas orthography have been assigned to various punctuation keys according to their frequency of use, estimated with a small sample of manually transcribed text (10 pages from Boas and Hunt (1921)).
- All accents are typed after the base character. Examples are shown below:
    - ä is typed *a then square bracket]*
    - k· is typed *k then slash /*
    - ō is typed *o then single quote ’*
    - â is typed *a then shift + comma ,*
    - ă is typed *a then shift + period .*
    - g̣ is typed *g then shift + square bracket]*
    - q´ is typed *q then option (alt key) + 1*

[1] https://keyman.com/developer/

[2] The keyboard package is available here: https://bit.ly/3Owysib.

- Other special characters are:
  - ᴇ is assigned to *semicolon ;*
  - ł is assigned to *square bracket [*
  - Ł is assigned to *shift + square bracket [*
  - ᵋ is assigned to *option (alt key) + e*
  - ᵘ is assigned to *option (alt key) + u*
  - ʟ is assigned to *option (alt key) + l*
- All changed punctuation keys can type their original value by holding down the Alt or Option key. For example, to get the original value of the square bracket [, type *Alt + [* (Windows) or *Option + [* (Mac).

A.2 Kwak'wala Transcription: Post-Completion Survey

In Section 5.2, we describe a user study to evaluate the utility of OCR and post-correction models on reducing the time and effort needed for manual transcription. After participants completed transcriptions tasks, we also asked them to fill out a survey to get subjective feedback on their experience with the tasks. Discussion and analysis of the answers from the survey is in Section 5.2.7. We provide a complete list of the questions asked in the survey here:

1. Were there specific tasks you found easier or more difficult to annotate?
2. Did you prefer typing the text from scratch or correcting predictions from a model? Why?
3. If you are a Kwak'wala language learner, did the annotation help your language learning? How?
4. Did the practice task help you become familiar with the keyboard?
5. After annotating a few pages, do you feel like you became faster at annotation?
6. Which do you feel is faster: typing from scratch or correcting predictions?
7. Any other feedback or thoughts on the task?

Bibliography

Antonios Anastasopoulos. *Computational tools for endangered language documentation*. University of Notre Dame, 2019. 1

Antonios Anastasopoulos and David Chiang. Leveraging translations for speech transcription in low-resource settings. In *Proc. INTERSPEECH*, 2018. 3.4

Antonios Anastasopoulos and Graham Neubig. Pushing the limits of low-resource morphological inflection. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 984–996, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1091. URL https://www.aclweb.org/anthology/D19-1091. 3.4, 7

Antonios Anastasopoulos, Christopher Cox, Graham Neubig, and Hilaria Cruz. Endangered languages meet modern nlp. In *Proceedings of the 28th International Conference on Computational Linguistics: Tutorial Abstracts*, pages 39–45, 2020. 1

Philip Arthur, Graham Neubig, and Satoshi Nakamura. Incorporating discrete translation lexicons into neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1557–1567, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1162. URL https://www.aclweb.org/anthology/D16-1162. 4.5

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015. 3.2, 4.1

Douglas Bates. Linear mixed model implementation in lme4. *Manuscript, University of Wisconsin*, 15, 2007. 5.2.6

Taylor Berg-Kirkpatrick, Greg Durrett, and Dan Klein. Unsupervised transcription of historical documents. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 207–217, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P13-1021. 1, 2.3, 2.4, 5.3

Judith Berman. George hunt and the kwak'wala texts. *Anthropological Linguistics*, 36(4):483–514, 1994. 5.1

Steven Bird. Decolonising speech and language technology. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3504–3519, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics. doi: 10.18653/v1/2020.coling-main.313. URL https://aclanthology.org/2020.coling-main.313. 7.2

Franz Boas. *The Social Organization and the Secret Societies of the Kwakiutl Indians: Smithsonian Institution. United States National Museum. By Franz Boas. With 51 Plates*. Washington: G.P.O., 1897. 5

Franz Boas. Sketch of the kwakiutl language. *American Anthropologist*, 2(4):708–721, 1900. 1, 2.2, 5

Franz Boas. *"Kwakiutl." Pp. 423–557 in Handbook of American Indian Languages, vol. 40.1, Bureau of American Ethnology Bulletin, edited by Franz Boas*. Washington: G.P.O., 1911. 5

Franz Boas. *Ethnology of the Kwakiutl*. Number v. 35, pt. 2 in Annual report. 1921. URL https://books.google.com/books?id=rO88AQAAIAAJ. 1, 2.2

Franz Boas. *Geographical Names of the Kwakiutl Indians*. New York: Columbia University Press, 1934. 5

Franz Boas and George Hunt. *Kwakiutl Texts*. Leiden, New York: E.J. Brill; G.E. Stechert & Co., 1902. 5

Franz Boas and George Hunt. *Ethnology of the Kwakiutl: Based on Data Collected by George Hunt*. Washington: G.P.O., 1921. 5, 5.3, A.1

Brenda H Boerger and Verna Stutzman. Single-event rapid word collection workshops: Efficient, effective, empowering. 2018. 7.2

Gina Bustamante, Arturo Oncevay, and Roberto Zariquiey. No data to crawl? monolingual corpus creation from PDF files of truly low-resource languages in Peru. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 2914–2923, Marseille, France, May

2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://www.aclweb.org/anthology/2020.lrec-1.356. 3.4

Rodrigo Cámara-Leret and Jordi Bascompte. Language extinction triggers the loss of unique medicinal knowledge. *Proceedings of the National Academy of Sciences*, 118(24), 2021. 1

Stanley F Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–394, 1999. 4.3.1

Xiaoxue Chen, Lianwen Jin, Yuanzhi Zhu, Canjie Luo, and Tianwei Wang. Text recognition in the wild: A survey. *ACM Computing Surveys (CSUR)*, 54(2):1–35, 2021. 1

Colette Craig. A constitutional response to language endangerment: The case of nicaragua. *Language (Baltimore)*, 68(1):17–24, 1992. 1

Hilaria Cruz and Joseph Waring. Deploying technology to save endangered languages. *arXiv preprint arXiv:1908.08971*, 2019. 1

Andrew M. Dai and Quoc V. Le. Semi-supervised sequence learning. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 3079–3087, Cambridge, MA, USA, 2015. MIT Press. a)

Angela Dean and Daniel Voss. *Design and analysis of experiments*. Springer, 1999. 5.4, 5.2.4

Samantha Disbray, Ben Foley, Shruti Rijhwani, and Meladel Mistica. Reading it right: A case study in pintupi-luritja. In *Digital Approaches to Multilingual Text Analysis*, February 2022. 7.1

Rui Dong and David Smith. Multi-input attention for unsupervised OCR correction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2363–2372, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1220. URL https://www.aclweb.org/anthology/P18-1220. 3, 3.3.1, 3.4, 4.5, 6.4, 7.2

Yuning Du, Chenxia Li, Ruoyu Guo, Xiaoting Yin, Weiwei Liu, Jun Zhou, Yifan Bai, Zilin Yu, Yehua Yang, Qingqing Dang, et al. Pp-ocr: A practical ultra lightweight ocr system. *arXiv preprint arXiv:2009.09941*, 2020. 1

Matthew Francis-Landau. Mfst: A python openfst wrapper with support for custom semirings and jupyter notebooks, 2020. 4.4.1

Yasuhisa Fujii. Optical character recognition research at google. In *2018 IEEE 7th Global Conference on Consumer Electronics (GCCE)*, pages 265–266. IEEE, 2018. 1

Yasuhisa Fujii, Karel Driesen, Jonathan Baccash, Ash Hurst, and Ashok C Popat. Sequence-to-label script identification for multilingual ocr. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 1, pages 161–168. IEEE, 2017. 2.4, 2.4.1, 5.3

Yunze Gao, Yingying Chen, Jinqiao Wang, and Hanqing Lu. Reading scene text with attention convolutional sequence modeling. *arXiv e-prints*, pages arXiv–1709, 2017. 1

Dan Garrette, Hannah Alpert-Abrams, Taylor Berg-Kirkpatrick, and Dan Klein. Unsupervised code-switching for multilingual historical document transcription. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1036–1041, Denver, Colorado, May–June 2015. Association for Computational Linguistics. doi: 10.3115/v1/N15-1109. URL `https://aclanthology.org/N15-1109`. 6

Lenore A Grenoble and Lindsay J Whaley. *Saving languages: An introduction to language revitalization*. Cambridge University Press, 2005. 1

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1154. URL `https://www.aclweb.org/anthology/P16-1154`. 3.2.2

Kai Hakala, Aleksi Vesanto, Niko Miekka, Tapio Salakoski, and Filip Ginter. Leveraging text repetitions and denoising autoencoders in ocr post-correction. *arXiv preprint arXiv:1906.10907*, 2019. 6.4, 7.2

Ken Hale, Michael Krauss, Lucille J Watahomigie, Akira Y Yamamoto, Colette Craig, LaVerne Masayesva Jeanne, and Nora C England. Endangered languages. *language*, 68(1):1–42, 1992. 1

Mika Hämäläinen and Simon Hengchen. From the paft to the fiiture: a fully automatic NMT and word embeddings method for OCR post-correction. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 431–436, Varna, Bulgaria, September 2019. INCOMA Ltd. doi: 10.26615/978-954-452-056-4_051. URL `https://www.aclweb.org/anthology/R19-1051`. 3.3.1, 3.4, 4.5

K David Harrison. *When languages die: The extinction of the world's languages and the erosion of human knowledge*. Oxford University Press, 2008. 1

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. Revisiting self-training for neural sequence generation. *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020, Conference Track Proceedings*, 2020. 4.2, 4.2

Pan He, Weilin Huang, Yu Qiao, Chen Change Loy, and Xiaoou Tang. Reading scene text in deep convolutional sequences. In *Thirtieth AAAI conference on artificial intelligence*, 2016. 1

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 690–696, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P13-2121. 4.3.3, 4.4.1

Nikolaus P Himmelmann. Documentary and descriptive linguistics. 1998. 1

Leanne Hinton. Language revitalization: An overview. *The green book of language revitalization in practice*, pages 3–18, 2001. 1

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997. 3.2.1

Chris Hokamp and Qun Liu. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1141. URL https://www.aclweb.org/anthology/P17-1141. 4.5

Richard W Howell. The classification and description of ainu folklore. *The Journal of American Folklore*, 64(254):361–369, 1951. 5

R Reeve Ingle, Yasuhisa Fujii, Thomas Deselaers, Jonathan Baccash, and Ashok C Popat. A scalable handwritten text recognition system. *arXiv preprint arXiv:1904.09150*, 2019. 2.4

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. The state and fate of linguistic diversity and inclusion in the NLP world. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6282–6293, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.560. URL https://www.aclweb.org/anthology/2020.acl-main.560. 1, 7.2

Katharina Kann, Ryan Cotterell, and Hinrich Schütze. Neural multi-source morphological reinflection. In *Proceedings of the 15th Conference of the European Chapter of the Association* *for Computational Linguistics: Volume 1, Long Papers*, pages 514–524, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/E17-1049. 3.4

Kyōsuke Kindaichi. *Ainu Jojishi Yūkara no Kenkyū [Research on Ainu Epic Yukar]*. Tōkyō: Tōkyō Bunko, 1931. 2.2

Okan Kolak and Philip Resnik. OCR post-processing for low density languages. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 867–874, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/H05-1109. 3.4

Michael Krauss. The world's languages in crisis. *Language (Baltimore)*, 68(1):4–10, 1992. 1

Amrith Krishna, Bodhisattwa P. Majumder, Rajesh Bhat, and Pawan Goyal. Upcycle your OCR: Reusing OCRs for post-OCR text correction in Romanised Sanskrit. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 345–355, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/K18-1034. URL https://www.aclweb.org/anthology/K18-1034. 3, 3.3.1, 1, 3.4

Fethi Lamraoui and Philippe Langlais. Yet another fast, robust and open source sentence aligner. time to reconsider sentence alignment? In *XIV Machine Translation Summit*, Nice, France, Sept. 2013. 3.2.3

Kimberley L. Lawson. *Precious fragments: First Nations materials in archives, libraries and museums*. PhD thesis, University of British Columbia, 2004. URL https://open.library.ubc.ca/collections/ubctheses/831/items/1.0091657. 5.1

Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, 2013. 4.2

Jindřich Libovický and Jindřich Helcl. Attention strategies for multi-source sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 196–202, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-2031. URL https://www.aclweb.org/anthology/P17-2031. 3.2, 3.4

Chu-Cheng Lin, Hao Zhu, Matthew R. Gormley, and Jason Eisner. Neural finite-state transducers: Beyond rational relations. In *Proceedings of the 2019 Conference of the North Amer-*

ican Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 272–283, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1024. URL https://www.aclweb.org/anthology/N19-1024. 4.5

Patrick Littell, Anna Kazantseva, Roland Kuhn, Aidan Pine, Antti Arppe, Christopher Cox, and Marie-Odile Junker. Indigenous language technologies in canada: Assessment, challenges, and successes. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2620–2632, 2018. 1, 1

Shangbang Long, Xin He, and Cong Yao. Scene text detection and recognition: The deep learning era. *International Journal of Computer Vision*, 129(1):161–184, 2021. 1

Jiří Martínek, Ladislav Lenc, and Pavel Král. Training strategies for ocr systems for historical documents. In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 362–373. Springer, 2019. 1

Austin Matthews, Graham Neubig, and Chris Dyer. Using morphological knowledge in open-vocabulary neural language models. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1435–1445, 2018. 7.2

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. Coverage embedding models for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 955–960, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1096. URL https://www.aclweb.org/anthology/D16-1096. 3.2.2

Mehryar Mohri. On some applications of finite-state automata theory to natural language processing. *Nat. Lang. Eng.*, 2(1):61–80, March 1996. ISSN 1351-3249. doi: 10.1017/S135132499600126X. URL https://doi.org/10.1017/S135132499600126X. 4.3.3

Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1):69–88, 2002. 4.3.2, 4.3.3

Christopher Moseley. Atlas of the world's languages in danger, 2010. URL http://www.unesco.org/culture/en/endangeredlanguages/atlas. 1

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*, 2017. 3.3.1, 4.4.1

Marianne Nicolson and Adam Werle. An investigation of modern kwak'wala determiner systems. *Victoria, BC: University of Victoria ms*, 2009. 5, 5.1

Kai Niklas. Unsupervised post-correction of ocr errors. *Master's thesis. Leibniz Universität Hannover*, 2010. 4.5

Matt Post and David Vilar. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1314–1324, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1119. URL https://www.aclweb.org/anthology/N18-1119. 4.5

Prajit Ramachandran, Peter Liu, and Quoc Le. Unsupervised pretraining for sequence to sequence learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 383–391, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1039. URL https://www.aclweb.org/anthology/D17-1039. a)

Jhonnatan Rangel. Challenges for language technologies in critically endangered languages. In *UNESCO International Conference Language Technologies for All (LT4All)*, 2019. 1, 1

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. Weighting finite-state transductions with neural context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 623–633, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1076. URL https://www.aclweb.org/anthology/N16-1076. 4.5

C. Rigaud, A. Doucet, M. Coustaty, and J. Moreux. ICDAR 2019 competition on post-OCR text correction. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1588–1593, 2019. 3, 4.5

Shruti Rijhwani, Antonios Anastasopoulos, and Graham Neubig. OCR Post Correction for Endangered Language Texts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, November 2020. 2, 3

Shruti Rijhwani, Daisy Rosenblum, Antonios Anastasopoulos, and Graham Neubig. Lexically-Aware Semi-Supervised OCR Post-Correction. *Transactions of the Association for Computational Linguistics*, 2021. 2, 4

Daisy Rosenblum, Shruti Rijhwani, Michayla King, Antonios Anastasopoulos, and Graham Neubig. Developing optical character recognition for kwak'wala. In *Proceedings of the Workshop on Computational Methods for Endangered Languages (ComputEL): Special Session*, 2022. 5

Diana Schackow. Documentation and grammatical description of yakkha, nepal. `https://elar.soas.ac.uk/Collection/MPI186180`, 2012. Accessed: 2020-02-02. 2.2

Diana Schackow. *A grammar of Yakkha*. Language Science Press, 2015. 2.4.1

Carsten Schnober, Steffen Eger, Erik-Lân Do Dinh, and Iryna Gurevych. Still not there? comparing traditional sequence-to-sequence models to encoder-decoder neural networks on monotone string translation tasks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1703–1714, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee. URL `https://www.aclweb.org/anthology/C16-1160`. 3, 3.2.2

Sarah Schulz and Jonas Kuhn. Multi-modular domain-tailored OCR post-correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2716–2726, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1288. URL `https://www.aclweb.org/anthology/D17-1288`. 2.3, 3.4, 4.5

Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1099. URL `https://www.aclweb.org/anthology/P17-1099`. 3.2.2, 3.2.2

Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304, 2016. 1

Miikka Silfverberg and Jack Rueter. Can morphological analyzers improve the quality of optical character recognition? In *Septentrio Conference Series*, pages 45–56, 2015. 7.2

Ray Smith. An overview of the tesseract ocr engine. In *Ninth international conference on docu-*

ment analysis and recognition (ICDAR 2007), volume 2, pages 629–633. IEEE, 2007. 1

Valentin I. Spitkovsky, Hiyan Alshawi, Daniel Jurafsky, and Christopher D. Manning. Viterbi training improves unsupervised dependency parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 9–17, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W10-2902. 2

Paolo Stomeo. *Racconti greci inediti di Sternatía*. La nuova Ellade, s.I., 1980. 2.2

Lindia Tjuatja, Shruti Rijhwani, and Graham Neubig. Explorations in transfer learning for ocr post-correction. In *Fifth Widening Natural Language Processing Workshop (WiNLP)*, November 2021. 6.4

Xiang Tong and David A. Evans. A statistical approach to automatic OCR error correction in context. In *Fourth Workshop on Very Large Corpora*, 1996. URL https://www.aclweb.org/anthology/W96-0108. 4.5

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–85, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1008. URL https://www.aclweb.org/anthology/P16-1008. 3.2.2

Lars Vögtlin, Manuel Drazyk, Vinaychandran Pondenkandath, Michele Alberti, and Rolf Ingold. Generating synthetic handwritten historical documents with ocr constrained gans. *arXiv preprint arXiv:2103.08236*, 2021. 1

David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts, USA, June 1995. Association for Computational Linguistics. doi: 10.3115/981658.981684. URL https://www.aclweb.org/anthology/P95-1026. 4.2, 4.3

Jingyi Zhang, Masao Utiyama, Eiichro Sumita, Graham Neubig, and Satoshi Nakamura. Guiding neural machine translation with retrieved translation pieces. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1325–1335, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1120. URL https://www.aclweb.org/anthology/N18-1120. 4.5

Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009. 4, 4.2, 4.3

Barret Zoph and Kevin Knight. Multi-source neural translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 30–34, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1004. URL https://www.aclweb.org/anthology/N16-1004. 3.2, 3.2.1, 3.4

Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin Dogus Cubuk, and Quoc Le. Rethinking pre-training and self-training. *Advances in Neural Information Processing Systems*, 33, 2020. 4.2