Lexically-Aware Semi-Supervised Learning for OCR Post-Correction

Shruti Rijhwani,[1] **Daisy Rosenblum,**[2] **Antonios Anastasopoulos,**[3] **Graham Neubig**[1]
[1]Language Technologies Institute, Carnegie Mellon University
[2]University of British Columbia
[3]Department of Computer Science, George Mason University
srijhwan@cs.cmu.edu, daisy.rosenblum@ubc.ca,
antonis@gmu.edu, gneubig@cs.cmu.edu

Abstract

Much of the existing linguistic data in many languages of the world is locked away in non-digitized books and documents. Optical character recognition (OCR) can be used to produce digitized text, and previous work has demonstrated the utility of neural post-correction methods that improve the results of general-purpose OCR systems on recognition of less-well-resourced languages. However, these methods rely on manually curated post-correction data, which are relatively scarce compared to the non-annotated raw images that need to be digitized. In this paper, we present a semi-supervised learning method that makes it possible to utilize these raw images to improve performance, specifically through the use of *self-training*, a technique where a model is iteratively trained on its own outputs. In addition, to enforce consistency in the recognized vocabulary, we introduce a *lexically-aware decoding* method that augments the neural post-correction model with a count-based language model constructed from the recognized texts, implemented using weighted finite-state automata (WFSA) for efficient and effective decoding. Results on four endangered languages demonstrate the utility of the proposed method, with relative error reductions of 15-29%, where we find the combination of self-training and lexically-aware decoding essential for achieving consistent improvements.[1]

1 Introduction

There is a vast amount of textual data available in printed form (Dong and Smith, 2018). In this paper, we address the task of digitizing printed materials that contain text in endangered languages, i.e., languages that are in danger of becoming extinct due to the dwindling numbers of native speakers and younger generations shifting to other languages. Printed materials in endangered languages come from various sources, including cultural and educational texts and linguistic documentation.

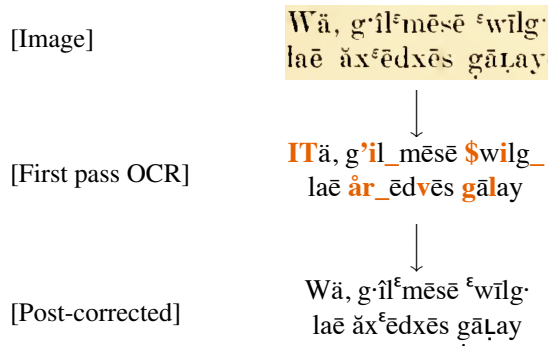

Figure 1: OCR post-correction on a scanned document that contains text in the endangered language Kwak'wala. The goal of post-correction is to fix the **recognition errors** made by the first pass OCR system.

Extracting text data from these documents is valuable for a multitude of reasons. Automatic digitization can aid language documentation, preservation, and accessibility efforts by archiving the texts and making them searchable for language learners and speakers. Further, 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). This challenge can be mitigated by converting printed materials in these languages to a machine-readable format.

Optical character recognition (OCR) systems can be used to produce digitized text, and recent work (Rijhwani et al., 2020) has demonstrated that *post-correction* improves the performance of existing general-purpose OCR systems on endangered languages (an example is in Figure 1). Most state-of-the-art OCR post-correction methods use neural

[1]Data and code are available at https://shrutirij.github.io/ocr-el/.

sequence-to-sequence models and rely on considerable resources such as a large number of manual transcriptions (Schnober et al., 2016; Rigaud et al., 2019) or substantial textual data to train a language model (Dong and Smith, 2018). To adapt these methods for the less-well-resourced endangered languages setting, Rijhwani et al. (2020) add translations and structural biases to the model.

However, even with such methods targeted to low-resource learning, post-correction performance is still dependant 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 paper, we propose 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** method for OCR post-correction (Section 4) 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 post-correction model. The self-training 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, we propose **lexically-aware decoding** (Section 5), 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). Our proposed 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.

We conduct experiments on four endangered languages: Ainu, Griko, Kwak'wala, and Yakkha. Our proposed method reduces the character and word error rates by 15%–29% over a state-of-the-art OCR post-correction method for endangered languages. We find that the *combination* of self-training and lexically-aware decoding is essential for achieving consistent improvements in performance.

2 Problem Formulation

Optical Character Recognition The OCR task involves generating a transcription of the text contained in an image. In this paper, we use existing OCR tools (detailed in Section 6.4) to obtain a *first pass transcription* for the images in our dataset. The first pass transcription is a text sequence of N characters, denoted as $\boldsymbol{x} = [x_1, \ldots, x_N]$.

OCR Post-Correction Even state-of-the-art OCR models are susceptible to making recognition errors (Dong and Smith, 2018). Errors are particularly frequent in the case of endangered languages because most off-the-shelf OCR tools do not directly support these languages and training a high-performance OCR system is challenging given the small amount of data that is typically available (Rijhwani et al., 2020). We use OCR post-correction to correct these errors and improve the quality of the transcription.

The post-correction model takes the first pass transcription $\boldsymbol{x}$ as input and generates the *corrected transcription*, a sequence of T characters denoted as $\boldsymbol{y} = [y_1, \ldots, y_T]$:

$$\boldsymbol{y} = \arg\max_{\boldsymbol{y'}} p_{\text{corr}}(\boldsymbol{y'}|\boldsymbol{x})$$

3 Base Model

As the base post-correction model, we use the model from Rijhwani et al. (2020): 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. We briefly describe the method here but refer readers to the original paper for details.

The OCR post-correction model takes the first pass transcription $\boldsymbol{x}$ as input, with the aim of predicting an error-free transcription $\boldsymbol{y}$. First, each character in the input sequence $\boldsymbol{x}$ is mapped to a vector representation using character embeddings. This forms a sequence of vectors, $\mathbf{x} =$

$[\mathbf{x}_1, \ldots, \mathbf{x}_N]$. The **encoder** is a character-level bidirectional LSTM (Hochreiter and Schmidhuber, 1997), which transforms $\mathbf{x}$ into a sequence of hidden state vectors $\mathbf{h} = [\mathbf{h}_1, \ldots, \mathbf{h}_N]$.[2]

The model's decoding process uses an **attention mechanism** to provide context from the encoder hidden states. At each decoding timestep t, the attention layer uses $\mathbf{h}$ to produce the context vector $\mathbf{c}_t$. The LSTM **decoder**, given $\mathbf{c}_t$, computes the output state $\mathbf{s}_t$ and subsequently the probability distribution $\mathbf{y}_t$ for generating the next character of the target sequence $\boldsymbol{y}$:

$$p(\mathbf{y}_t) = \text{softmax}(\mathbf{W}\mathbf{s}_t + \mathbf{b}) \quad (1)$$

Rijhwani et al. (2020) adapt the encoder-decoder model above for **low-resource post-correction** by adding pretraining and three structural biases:

- **Diagonal attention loss**: OCR post-correction is a monotonic sequence-to-sequence task. Hence, the attention weights are expected to be higher closer to the diagonal – adding attention elements off the diagonal to the training loss encourages monotonic attention (Cohn et al., 2016).
- **Copy mechanism**: The copy mechanism enables the model to choose between generating a character based on the decoder state (Equation 1) or copying a character directly from the input sequence $\boldsymbol{x}$ by sampling from the attention distribution (Gu et al., 2016; See et al., 2017).
- **Coverage**: The coverage vector keeps track of attention weights from previous timesteps. It is used as additional information when computing $\mathbf{c}_t$ and is added to the training loss to discourage the model from repeatedly attending to the same character (Mi et al., 2016; Tu et al., 2016).

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 $\mathcal{L}$) is a combination of cross-entropy loss along with the diagonal attention loss and the coverage loss from the structural biases. Inference with a trained model is performed using **beam search**.

[2] Rijhwani et al. (2020) incorporate translations into the model with a multi-source encoder. We omit this from our formulation, considering applicability to texts without available translations. However, adding an encoder into our framework remains straightforward and can be used if translations exist.

In the following sections, we use the method described above as a base model for our proposed semi-supervised learning technique for OCR post-correction. Given the minimal manually transcribed data we have in endangered languages, our approach 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.

4 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).

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. 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 the initial OCR post-correction model f_θ (trained on D) to 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. Train the model f_θ on pseudo-training sets U and S ("pseudo-train").

[3] In our dataset, the source and target data instances are either at the line-level or the sentence-level (see Section 6.1).

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

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).

5 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

[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.

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.

5.1 Count-Based Language Model

From the self-training method in Section 4, 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 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.

5.2 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 1), so in order to leverage word frequency in the decoding process, we convert the count-based word-level LM described in Section 5.1 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 2(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 2(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 2(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 5.1) 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-

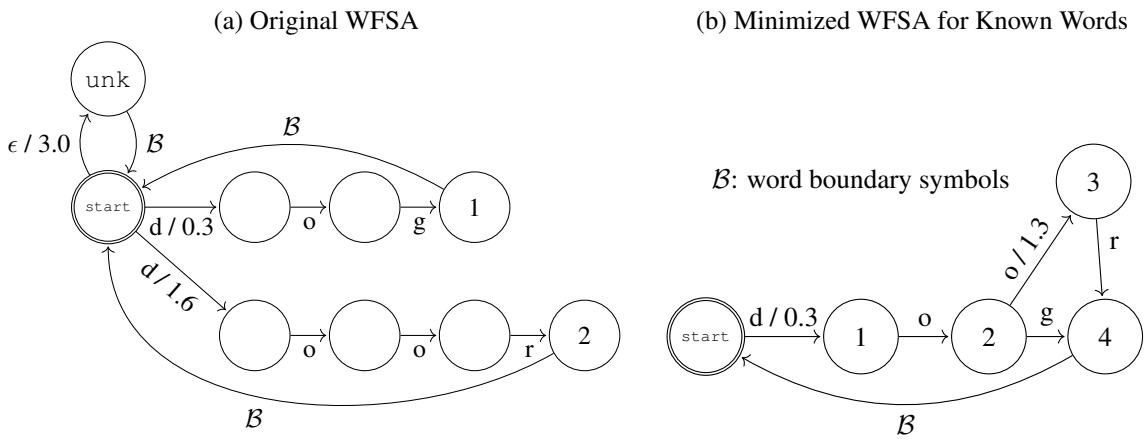

Figure 2: The (a) WFSA and (b) minimized WFSA we construct, for a hypothetical language model with a two word vocabulary: $P(\text{dog}) = 0.75$; $P(\text{door}) = 0.2$; $P(\text{<unk>}) = 0.05$. The transition weights are negative log probabilities. In (b), for simplicity, we show only the known word states after determinization and minimization.

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).

5.3 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 2(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 towards the start state of the 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 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

[5] We use $n = 6$ in this paper. 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.

minimized WFSA (see Figure 2(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:

$$\text{score}_{\text{lex}}(y_t) = \min(\text{known}(y_t), \text{unk}(y_t)) \quad (2)$$

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 2). 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.

5.4 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 1). We also compute $\text{score}_{\text{lex}}(y_t)$, which is a negative log probability, as defined in Equation 2. The final probability of predicting y_t is obtained through linear interpolation between these two scores,[6] weighted by a hyperparameter λ:

$$p(y_t) = (1 - \lambda) \cdot p_{\text{lstm}}(y_t) + \lambda \cdot p_{\text{lex}}(y_t) \quad (3)$$

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

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

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 $p_{\text{lex}}(y_t)$.

6 Experiments

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

6.1 Datasets

We use the OCR post-correction dataset from Rijhwani et al. (2020) which contains transcribed documents in three endangered languages: Ainu, Griko, and Yakkha. Additionally, in this paper, we create a similar dataset in the endangered language Kwak'wala. We describe the datasets below, including the sizes of the gold transcribed and unannotated sets we use for semi-supervised training:

Ainu (`ain`) is a severely endangered language from northern Japan. The dataset contains pages from a book of Ainu epic poetry (Kindaichi, 1931). The Ainu text is written in the Latin script. The dataset contains 816 manually transcribed lines as well as 7,646 lines without gold transcriptions.

Griko (`grk`), an endangered Greek dialect spoken in southern Italy, is written with a combination of Latin and Greek alphabet. The document in the dataset is a book of Griko folk tales (Stomeo, 1980). There are 807 and 3,084 sentences with and without gold transcriptions, respectively.

Yakkha (`ybh`) is an endangered language spoken in Nepal and is written in the Devanagari script. The dataset contains transcriptions of three children's books (Schackow, 2012). 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 ($\approx$ 30 sentences: see Section 6.2 for data splits).

Kwak'wala (`kwk`) is spoken on Northern Vancouver Island, nearby small islands, and the opposing mainland. The language is severely endangered, with estimates of $\approx$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 vow-

els. Several writing systems exist: including the U'mista, Liq'wala, and Boas orthographies.

The Boas orthography (Boas, 1900) was developed by anthropologist Franz Boas. It 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 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.

6.2 Experimental Setup

Data Splits We follow Rijhwani et al. (2020) and perform 10-fold cross-validation for all experiments. For each language, the gold-transcribed data is split into 10 segments, and for each cross-validation fold, eight segments are used for training, one for validation, and one for testing.

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 (Berg-Kirkpatrick et al., 2013; Schulz and Kuhn, 2017). CER is the character-level edit distance between the predicted text and the corresponding gold transcription, divided by the total number of characters in the gold transcription. WER is similar but is calculated at the word-level. For readability, we report CER and WER as percentages for all experiments.

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

- FIRST-PASS: To obtain the first pass OCR transcription, we experiment with two existing OCR systems: Google Vision (Fujii et al., 2017) and Ocular (Berg-Kirkpatrick et al., 2013).

  For each language, we choose the best performing first pass system, the details of which are in Section 6.4. We use Ocular for Kwak'wala and Google Vision for Ainu, Griko, and Yakkha.

- BASE: The current state-of-the-art in OCR post-correction for endangered language texts (Rijhwani et al. (2020); described in Section 3).

- SEMI-SUPERVISED: Our proposed method as described in Sections §4 and §5.

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. Following Rijhwani et al. (2020), results reported are the average of five randomly seeded runs (i.e., five runs for each of the 10 cross-validation folds).

6.3 Main Results

Table 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 1 presents the best models based on the validation set WER. Extensive analysis of these factors is in Section 6.6.

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

Model	% Character Error Rate				% Word Error Rate			
	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 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.

OCR System	% Character Error Rate				% Word Error Rate			
	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: 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.

6.4 First Pass OCR Systems

We experiment with two existing OCR systems to obtain a first pass transcription on our dataset. The first of these is the Google Vision system (Fujii et al., 2017; Ingle et al., 2019). This large-scale OCR model supports 60 languages in 27 scripts – these are primarily higher-resourced languages and do not include our target endangered languages.

The second system is Ocular (Berg-Kirkpatrick et al., 2013). Ocular uses a generative model to transcribe scanned documents: the model generates the image by learning the font of the document. Ocular relies on a character n-gram language model trained on the target language. We initialize the LM with the small number of gold-transcribed pages in our dataset. For this, we use the 10-fold cross-validation setup described in Section 6.2: we use the training segments to train the Ocular LM and the test segment to evaluate OCR performance. The font model has parameters to learn the shape of each character in the LM vocabulary. After initialization, the parameters are updated in an unsupervised manner with EM until convergence.

Results are presented in Table 2. 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.[7] On the other hand, the performance is much worse on Kwak'wala since the system has not been trained on the Boas orthography.

We find that the performance on Kwak'wala is considerably better with the Ocular system because the LM 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 Google Vision, likely due to the minimal data available for training it. Moreover, the performance is correlated with the *word overlap* between test data and the data used for training the LM, demonstrating Ocular's reliance on a strong language model – the word overlap is 73% for Griko, 56% for Kwak'wala, and 48% for Ainu.

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 written. More specifically, when a vowel diacritic is applied to a consonant, the characters are combined: e.g., क + ा = का. In Unicode, this is represented by two characters "क "

[7] See Rijhwani et al. (2020) for a more detailed analysis.

Known Word Model	Unknown Word Model	% Character Error Rate				% Word Error Rate			
		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 3: 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.

and "◌ɪ", where the dotted circle is the *character combination marker* in the Unicode Standard.[8]

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 "◌ɪ" 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.

6.5 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 3). 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 5.1, 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 3. 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.

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

Lang. Code	LM Coverage	Known		Unknown	
		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: 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.

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 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.

6.6 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 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 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

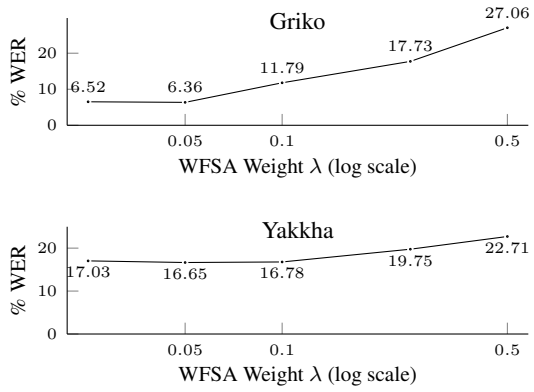

Figure 3: The weight of the WFST during joint decoding can affect word error rate (sometimes significantly, as in Griko; top). All other hyperparameters are kept equal and correspond to the best systems in each language.

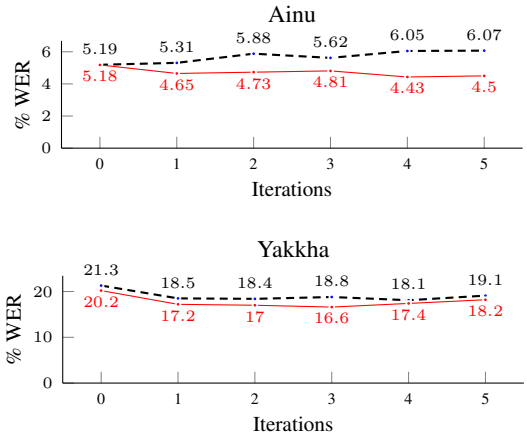

Figure 4: Integrating lexically-aware decoding through interpolation with a WFSA (red lines) aids self-training in improving WER across iterations. Black dashed lines correspond to self-training without lexical decoding.

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.

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 3). Specifically, in the case of Griko, we find that the value of this hyperparameter can significantly affect performance. As shown in Figure 3, 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 3, where increasing λ does not affect performance as much as in Griko.

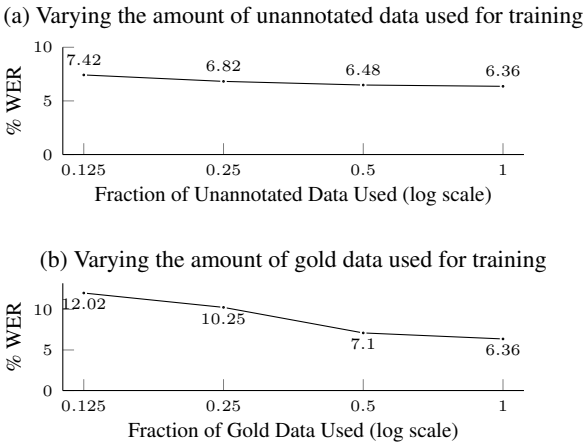

Figure 5: Even a small amount of unannotated data is useful for our semi-supervised method, improving WER over BASE (WER=7.51) in (a). Varying the size of gold-annotated data has a stronger effect on post-correction performance in (b). Results are shown with Griko.

Self-Training Iterations The evolution of WER across 5 self-training iterations for Ainu and Yakkha is shown in Figure 4. 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 Figure 5 (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 6.4). Figure 7 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 perfor-

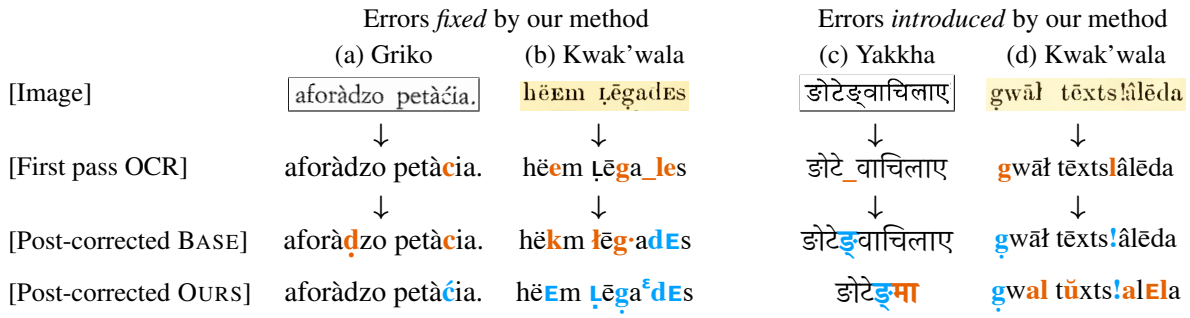

Figure 6: Our post-correction model **fixes** many of the first pass OCR **errors** that the base model does not fix such as (a) and (b). In rare cases, our method introduces **errors** into the transcription such as (c) and (d).

mance 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 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 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 6 (c) and (d), 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.

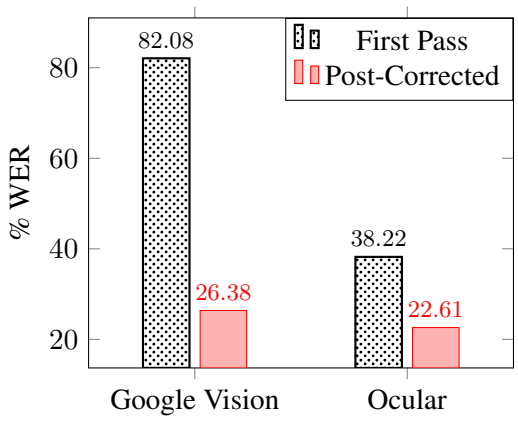

Figure 7: Our post-correction model significantly improves recognition accuracy over different first pass OCR systems that have varied error rates (Google Vision and Ocular). Results are shown with Kwak'wala.

7 Related Work

OCR post-correction is well-studied in the high-resource setting, particularly for English. Recent methods primarily use neural encoder-decoder models (Dong and Smith, 2018; Rigaud et al., 2019; Hämäläinen and Hengchen, 2019). There has been relatively little work on lower-resourced languages. Kolak and Resnik (2005) present a probabilistic edit distance model for post-correction on Cebuano and Igbo, and Krishna et al. (2018) use a sequence-to-sequence model with a copy mechanism for improved performance on Romanized Sanskrit OCR.

While existing neural post-correction methods 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 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 propose in this paper.

8 Conclusion

Digitization at scale for documents in under-represented languages is a promising avenue towards tackling one aspect of their marginalization, the lack of data. With this work, we take a step towards better digitization for extremely data-scarce scenarios. We develop a semi-supervised method that combines self-training with lexically-aware decoding, reducing error rates by up to 29% over a state-of-the-art OCR post-correction model on four typologically diverse endangered languages.

In future work, we plan to expand our method to take advantage of additional outputs of the language documentation process. For example, documentary linguists typically collect word lists (which range from lists of common words like the Swadesh lists (Swadesh, 1955) to domain-specific vocabularies). Using such word lists within the lexically-aware decoding framework could further improve performance and enable the application of our technique to even lower-resourced languages.

Additionally, the improvements we achieve through semi-supervised learning are potentially orthogonal to the improvements Rijhwani et al. (2020) achieve by incorporating information from translations of the target text. As future work, we plan to investigate the combination of these two approaches in an attempt to utilize all available sources of information to improve performance.

Finally, while using a character-level n-gram LM improves performance on unknown words, it does not explicitly utilize morphological structure to generate unseen inflections of words. In the future, we plan to incorporate morphological analysis during post-correction decoding, which will be helpful for morphologically rich endangered languages.

Acknowledgments

This work has been supported by grant PR-276810-21 (*"Unlocking Endangered Language Resources"*) from the National Endowment for the Humanities and grant 1761548 (*"Discovering and Demonstrating Linguistic Features for Language Documentation"*) from the National Science Foundation. Any views, findings, conclusions or recommendations expressed in this publication do not necessarily represent those of the National Endowment for the Humanities.

The authors would like to thank Siddharth Dalmia, Deepak Gopinath, Samridhi Choudhary, the anonymous reviewers, and the Action Editors for feedback on the paper. Shruti Rijhwani would also like to acknowledge support from the Bloomberg Data Science Ph.D. Fellowship.

References

Philip Arthur, Graham Neubig, and Satoshi Nakamura. 2016. 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. Association for Computational Linguistics.

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

Taylor Berg-Kirkpatrick, Greg Durrett, and Dan Klein. 2013. 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. Association for Computational Linguistics.

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

Franz Boas. 1921. *Ethnology of the Kwakiutl*. Number v. 35, pt. 2 in Annual report.

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

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. In *Proceedings of the 2016 Conference of the North American Chapter of the*

Association for Computational Linguistics: Human Language Technologies, pages 876–885, San Diego, California. Association for Computational Linguistics.

Andrew M. Dai and Quoc V. Le. 2015. 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. MIT Press.

Rui Dong and David Smith. 2018. 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. Association for Computational Linguistics.

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

Yasuhisa Fujii, Karel Driesen, Jonathan Baccash, Ash Hurst, and Ashok C Popat. 2017. 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.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. 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. Association for Computational Linguistics.

Mika Hämäläinen and Simon Hengchen. 2019. 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. INCOMA Ltd.

Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. 2020. 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*.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. 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. Association for Computational Linguistics.

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

Chris Hokamp and Qun Liu. 2017. 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. Association for Computational Linguistics.

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

Pratik Joshi, Sebastin Santy, Amar Budhiraja, Kalika Bali, and Monojit Choudhury. 2020. 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. Association for Computational Linguistics.

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

Okan Kolak and Philip Resnik. 2005. 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. Association for Computational Linguistics.

Amrith Krishna, Bodhisattwa P. Majumder, Rajesh Bhat, and Pawan Goyal. 2018. 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. Association for Computational Linguistics.

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

Chu-Cheng Lin, Hao Zhu, Matthew R. Gormley, and Jason Eisner. 2019. Neural finite-state transducers: Beyond rational relations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 272–283, Minneapolis, Minnesota. Association for Computational Linguistics.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. 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. Association for Computational Linguistics.

Mehryar Mohri. 1996. On some applications of finite-state automata theory to natural language processing. *Nat. Lang. Eng.*, 2(1):61–80.

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

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. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

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

Matt Post and David Vilar. 2018. 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. Association for Computational Linguistics.

Prajit Ramachandran, Peter Liu, and Quoc Le. 2017. 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. Association for Computational Linguistics.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. 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. Association for Computational Linguistics.

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

Shruti Rijhwani, Antonios Anastasopoulos, and Graham Neubig. 2020. OCR Post Correction for Endangered Language Texts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5931–5942, Online. Association for Computational Linguistics.

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

Carsten Schnober, Steffen Eger, Erik-Lân Do Dinh, and Iryna Gurevych. 2016. 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. The COLING 2016 Organizing Committee.

Sarah Schulz and Jonas Kuhn. 2017. 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. Association for Computational Linguistics.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. 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. Association for Computational Linguistics.

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

Morris Swadesh. 1955. Towards greater accuracy in lexicostatistic dating. *International journal of American linguistics*, 21(2):121–137.

Xiang Tong and David A. Evans. 1996. A statistical approach to automatic OCR error correction in context. In *Fourth Workshop on Very Large Corpora*.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. 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. Association for Computational Linguistics.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Jingyi Zhang, Masao Utiyama, Eiichro Sumita, Graham Neubig, and Satoshi Nakamura. 2018. 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. Association for Computational Linguistics.

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

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