Jewett Digitization

Madi Sanchez-Forman
Computer Science and Mathematics
University of Puget Sound
msanchezforman@pugetsound.edu

Seung Park
Computer Science
University of Puget Sound
separk@pugetsound.edu

Ben McAuliffe
Computer Science
University of Puget Sound
bmcauliffe@pugetsound.edu

Emilee Oquist
Computer Science
University of Puget Sound
eoquist@pugetsound.edu

ABSTRACT

This paper outlines a project that attempts to transcribe Stanley Jewett's cursive handwritten diaries using Tesseract's optical character recognition (OCR) engine. The main stages of the project pipeline were pre-processing, training, post-processing, and testing of the model. Python's OpenCV and NumPy packages were used for pre-processing. Training was compounded on top of a pre-trained Tesseract OCR Engine, and the post-processing stage involved using a custom dictionary and the SpaCy natural language processing model to correct inaccuracies. The model was tested by making predictions on new images and comparing the output to human-transcribed text. After completing some sample tests, the character error rate (CER) was approximately 18% without post-processing. After, it achieved a 3% CER on one line, but the average edit distance was 11.6 characters. Future work involves expanding the data-set, improving pre-processing for complex pages, exploring alternative OCR methods, and enabling public access to Stanley Jewett's notes.

KEYWORDS

Cursive, Transcription, Optical Character Recognition, Hidden Markov Model, Convolutional Neural Network, Recurrent Neural Network, Long-Short Term Memory, CNN-N-Gram, Pre-Processing, Post-Processing, Tesseract

1 INTRODUCTION

Stanley Gordon Jewett (1885-1955) was a preeminent naturalist and field biologist in the Pacific Northwest [1]. He was employed by the Bureau of Biological Survey and the US Fish and Wildlife Service from 1911 - 1949. He was the head of predator control for Washington and Oregon from 1915-1934, the first supervisor of the Malheur National Wildlife Refuge (1935) and was the first Pacific Flyway biologist. He was meticulous and took thousands of pages of notes documenting his actions, collections, and observations, with an emphasis on birds and mammals. After Jewett's death, his day-books and some of his field notes were donated to the University of Puget Sound, where they remained in the archives, until Professor Peter Wimberger, Director of the Slater Museum at Puget Sound, rediscovered them and decided to make a consummate effort at digitizing the notes. Much of the natural state of the Pacific Northwest is undocumented during Jewett's active time period, and his notes could be of significant importance to many biologists and ecologists working in the region.

Similar to many people in the early 1900s, Jewett wrote in a unique brand of near-illegible cursive. In recent times, cursive has been phased out of the mandatory curriculum, so fewer people are proficient at reading and writing in cursive. Human transcription is also a very time-consuming process. On top of this, the pool of people capable of accurately transcribing Jewett's field notes is slim as it requires some amount of familiarity with outdated species' common and scientific names. For these reasons, merely uploading scans of Jewett's notebooks online would not be of lasting benefit. The primary motivations for transcribing his writings into text are not only to increase legibility but also to allow that data to be accessible. These were the driving forces that led professor Wimberger to approach University of Puget Sound's Computer Science department to propose the creation of an OCR engine to automate this archival digitization.

OCR engines have been increasing in prevalence over the past few years with most major tech companies incorporating live text and voice recognition into their software. However, most of these tools can only be used on typed font or clear print handwriting, and even then it is not without their fair share of errors. Modern-day text recognition software performs well enough, however cursive has always been a challenge with few attempts made in the past thirty years.

We took on the project to see if we would be able to accurately transcribe Jewett's journals by further training on Google's Tesseract with a custom built data-set. We believe this goal to be possible if and only if we are able to create a large enough data-set. However, the few months over which this project took place were simply not enough time to manually build a data-set of the scale needed. Nevertheless, we endeavored to achieve the highest accuracy rate that we could using pre-processing, model training, and post-processing.

2 BACKGROUND AND RELATED WORK

Many different types of architectures have been used in approaching handwritten recognition modeling. One approach is using Hidden Markov Models (HMM), which are probabilistic frameworks that handle data as a sequence of observations over time. HMM's have been successful with problems such as automatic speech recognition. Therefore, people have also applied these models to the handwriting recognition problem with limited success. More recently, HMM's have been replaced by many other types of architectures. Another approach is using Convolutional Neural Networks (CNN), which use convolutional layers that apply filters to transform the data. CNN's are useful for finding patterns and objects in complex

data such as images. Another architecture that has proved to be useful are Recurrent Neural Networks (RNN). RNN's are similar to CNN's, but they are designed to analyze temporal or sequential data, and have a memory mechanism that allows them to reuse activation functions from other data points in the sequence in order to generate the next output in a series. Because of this, RNN's are commonly used for speech recognition, language modeling, and handwriting recognition.

El-Yacoubi et al. [2] used an HMM to recognize cursive handwriting. The first step they use consists of pre-processing, in which they apply baseline slant normalization, character slant normalization, and lower-case letter area normalization. The next step was to segment the words into characters using an explicit segmentation algorithm with a high number of segmentation points, and the last step before predicting the words is feature extraction. There are three feature sets, and these feature sets all extract features that will be able to be processed as sequential data that can then output a prediction for which letter and word it is on the image.

Many researchers have found that using CNN's with the help of other techniques have proved to be useful in handwriting recognition. For example, Poznanski et al. [3] used a CNN-N-Gram model in which they utilized separate and parallel fully connected layers that led to a separate group of attribute predictions. Those attribute predictions were then divided into groups of unigrams, bigrams, or trigrams to learn the approximate position of the N-gram in the word. Researchers have also found that Long Short-Term Memory neural networks (LSTM) are also useful in handwriting recognition. LSTM's are a type of RNN that contain feedback to, in a sense, remember the important information. Stuner et al. [4] use a cascade of hundreds of LSTM-RNN's and reject the predicted word during decoding if it is not close enough to a word in the lexicon. Lastly, they then apply viterbi lexicon decoding on the rejected words.

Some researchers have decided to combine both CNN and RNN architecture in order to utilize the benefits of both types of neural networks. Sueiras et al. [5] uses the method of a sliding window over the input image and is given over to a convolution layer that extracts the most important features. It is then given to an encoder-decoder Bi-directional LSTM. The last part of their architecture is an attention mechanism that helps to focus on specific parts of the encoder output features that are relevant to identifying specific letters. Another example of this type of model was implemented by Dutta et al. [6] that used a CNN-RNN Hybrid Network. They tested their network with a variation of different layers. The final architecture of their model consisted of an input image, a spatial transformer network layer, residual convolutional blocks followed by stacked Bi-directional LSTM's, and finally, a connectionist temporal classification layer that then transcribes the labels. They utilized the image slant and slope normalization technique to pre-process their data and pre-trained their network on synthetic data from the IIIT-HWS data-set that was created to accommodate the lack of real-life handwritten data available to them. The next step was to distort and transform existing data-sets such as the IAM data-set and RIMES data-set with various data augmentation techniques to continue training on their network.

3 IMPLEMENTATION AND METHODS

This project pipeline consisted of multiple steps in the training and testing process. Figure 1 illustrates this process. We take scans of pages from a journal, and then pre-process the images. First, we convert to grayscale, then binarize the image, and lastly remove any dotted lines from the image. Next, we create the files needed to train Tesseract 5 by splitting the page images into individual lined tiff files, box files, and ground truth text files. The next part of the training pipeline is to train Tesseract on Jewett's handwriting with this data-set. We then use the trained model to recognize the text on the pre-processed testing images and then run it through post-processing which attempts to correct words that are misspelled.

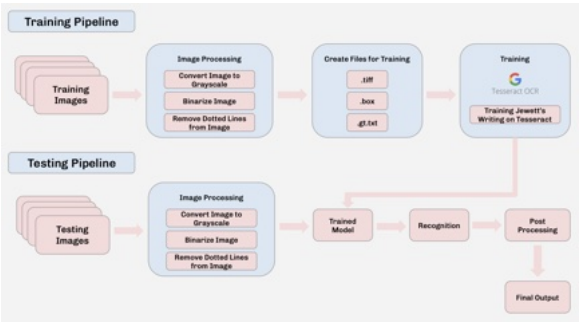

Figure 1: Overview of the project pipeline. The various important details of the project pipeline is illustrated such as image processing, the files required to train the model, and post-processing

3.1 Past Implementations

At the beginning of the capstone project, our team was excited to create an OCR from scratch using TensorFlow and Keras which we believed would allow us to customize the model to our specific needs. We spent several weeks researching deep learning models, determining which architecture pipeline works best, and considering various approaches to noise removal. We started out by determining how to pre-process our images. We needed to find a way to segment our text in order to separate words from each other using a kind of contour detection and then the next step was to find a way to separate individual letters from each other, despite the letters being connected components. Because none of us had prior experience in artificial intelligence or neural networks, we found ourselves struggling to keep up with the steep learning curve.

To remove lines and other noise, we implemented the horizontal projection profile and A* algorithm in the early stages of the project. However, we soon realized that building a high-quality OCR system from the ground up was much more complex than we had anticipated. The first half of our project timeline consisted of scanning physical journals and field notes, manually transcribing the contents, verifying those transcriptions, pre-processing those images, and attempting to discern the building blocks for a typical OCR.

The data-set collection and labeling were incredibly tedious, and Peter Wimberger, the professor advising the project and the most

adept at interpreting the contents of the journals, was critical to the transcription process, but he could only work so fast especially given his other responsibilities.

Building our own OCR was just not practical given the time constraints of our project. Our professor, America Chambers, pointed out that it would take at least a year if not more to develop a reliable OCR system from scratch. This was around halfway mark of our total time to work on the project. Due to this advice, we switched to the pre-trained Tesseract OCR engine letting us make the most of the work that had already been done by others in the field. This would also allow us to develop our own customizations to the model that would bring us closer to our goal of transcribing a specific person's handwriting. Following our switch to Tesseract, our team also began to work with jTessBoxEditor. This program handled the creation of bounding boxes used for training Tesseract. Using jTessBoxEditor's interface made it easier for us to create and move virtual frames around a letter on a pre-processed page as well as label each letter inside. Box data with the labels would be fed to Tesseract and it would learn to associate box shapes and their labels to novel data. When we initially tried to fine-tune the pre-trained OCR to recognize Jewett's handwriting, we found that the results were not satisfactory.

After some digging, one of our team members uncovered a brief reference stating that Tesseract is used on line-by-line images instead of the page data we were using. This meant, once again, drastically changing the data we were working with. All of the bounding box data files we had created were now rendered useless because it required pixel coordinates which would be altered when the image was split into line segments. This was near the end of the project before we would be forced to wrap up. With these last-minute changes, the amount of training data we were able to prepare for our model became much smaller than we would have liked. With the data now formatted properly, we were able to train Tesseract with much better results than before. With the model architecture solidified and achieving a somewhat successful iteration of the model, the proceeding sections will discuss more in-depth about each stage in the completed model.

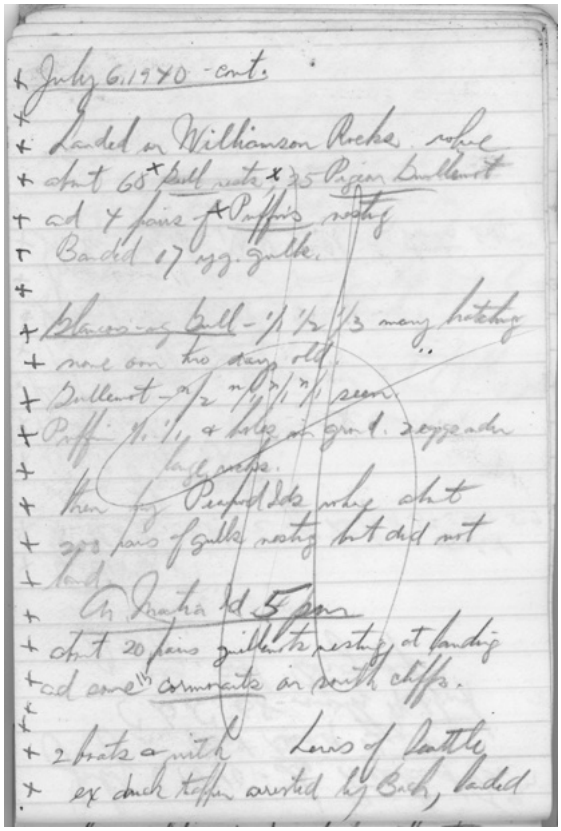

Figure 2: Large 'H' marking across messy page.

3.2 Pre-Processing

Often in projects attempting to transcribe offline text, the first step in the process is image pre-processing. The goal of image pre-processing is to isolate the characters on a page, such that excess noise and markings are not mistaken for characters. Since our goal was to transcribe diaries with highly varying formats, the pre-processing that worked for one journal was not applicable to the next. Therefore, we broke up the diaries into two categories: messy and clean. 'Messy' notebooks were characterized by the amount of excess noise on the pages. This often meant diaries written in pencil fell into this category, as over time these diaries were handled frequently causing smudging across the page. Messy notebooks were also those that contained a large 'H' mark shown in figure 2, or large amounts of lines or scribbles.

Both categories followed similar steps, but messier diaries required an extra step. For this reason we will start by illustrating the process of isolating characters on cleaner notebooks. All images were pre-processed using the Python packages, OpenCV (Open Source Computer Vision Library) and NumPy. OpenCV is an open source library consisting of more than 2,500 computer vision and machine learning algorithms. These algorithms can be used for many different problems including facial recognition, editing images, and searching through databases by image. It has a growing community and is used by the government, tech companies, and for research. NumPy is one of the most widely used Python packages. It is used for anything from simple mathematical operations, to operating on large multi-dimensional arrays and matrices. This makes NumPy ideal for representing our images as arrays of pixels that can then be operated on with OpenCV. Figure 3 illustrates the four basic stages of pre-processing a line from a clean notebook.

Initially, we start with a high quality scan of a page that is then converted to gray scale. The next stage is the most important, and it is known as image binarization or thresholding. This is where each pixel in the image is converted to white if the gray value is greater than or equal to some threshold and converted to black if the value is less than the threshold. Therefore each pixel in the image is now either pure white or black. At one point, we stopped the pre-processing here, but the dotted lines from these journals were getting recognized as periods. To remedy this, we had to filter them out. This is what we did in the final stage of pre-processing. After trial and error, we found that the most accurate way to filter out these dots without losing too much of the text is to filter out connected components of the image by area. We used OpenCV's

Figure 3: Pre-processing stages for clean notebooks. Line reads "Storm continues"

connected components function to find all connected components on an image, and filter out pixels less than a certain size. This allowed us to remove these lines without significant alterations to the text.

As previously stated, messier notebooks followed this same four-stage process, but with one added noise removal step. To perform this extra step, Gaussian Blur was applied to the entire image with a kernel size of 3. This allowed each pixel on the page to become a weighted average of the surrounding pixels, which muted excessively noisy pixels. After the blurring, the image was binarized again in order to further eliminate excess noise. These journals posed a larger challenge due to the smudging. When binarizing the images, smudging causes some pixel values to be lighter than ones surrounding them, leading to the loss of some important pixels.

3.3 Training

Instead of creating our own neural network, we opted to train on top of a pre-trained model. We chose Tesseract, which is a pre-trained OCR Engine that is distributed under the Apache 2.0 license, and in 2005, HP released Tesseract as an open source tool. Tesseract does not have a graphical user interface, so programmers must train Tesseract using the command line.

Figure 4 illustrates Tesseract's internal architecture. Tesseract uses a series of image processing techniques to prepare the image for text detection and recognition. Image processing consists of the Page Layout Analysis, Line Finding, Baseline Fitting, Fixed Pitch Detection and Chopping, and Proportional Word Finding. The next step is word recognition where chop points are found and then broken characters are then reconstructed by associating broken characters as one if the potential chops were not good enough. This then leads to the classification portion where the class pruner creates a shortlist of characters that the word may contain. Then each potential character is compared to each of the characters in the shortlist which are represented as a bit vector of prototypes of the given class that it might match known as configurations. Finally, it goes through a list of words that each predicted word may match with, and then it outputs the final prediction.

Tesseract has already been trained on over 400,000 text lines spanning about 3,500 different fonts. Tesseract also has models for over 100 different languages. To further train Tesseract on Stanley Jewett's handwriting, Jewett's handwriting was treated as another font.

Three files per datum were required to run the training: a Tiff file, a Box file, and a Ground Truth text file. A Tiff file is the actual lines image, a box file contains the coordinates, widths, heights, and character mappings of each letter on the line. In total, 348 line images were created, and after training on varying amounts of epochs the best accuracy on training and testing we achieved was 3,000 epochs.

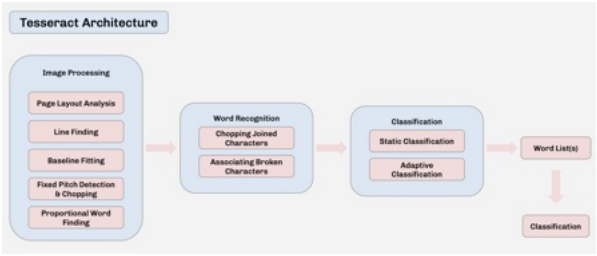

Figure 4: Overview of Tesseract's architecture.

3.4 Post-Processing

After training, the output of the model had very few entirely correctly spelled words, which created the need for post-processing to attempt to correct as much of the misspellings as possible. The idea of this form of post-processing was born originally from the desire to be able to find the precise Levenshtein distance, that is the minimum number of single character edits to get from one spelling to another, from the model output and the actual transcription. The hope was that the model output would closest resemble the actual word given a limited dictionary of words to choose from.

The limited dictionary is not a perfect idea and leads to a large flaw, but for the time we had it was the best tool for the job. The limited dictionary was made from every word that we know Jewett had written combined with a list of all bird and mammal scientific names that we would be likely to come across in the future. The first part we were able to build by scanning over all of the current transcriptions of pages, while the second was put together by Professor Wimburger.

There were 3 distinct attempts at post-processing. The first was based purely on the shortest absolute distance between the output word and the dictionary. We noticed that the model had a stronger tendency to get the first letter in a word correct as these letters were often more distinct in the writing. In an effort to increase post-processing accuracy, we put an added weight to the first letter if it was matching a word in the dictionary such that it would slightly

decrease the edit distance. The final optimization was to include SpaCy, an NLP model, to help predict the next word in the sequence.

4 RESULTS

Throughout our experiences working with various texts written by Jewett, we realized how critical and delicate the pre-processing stage is for maintaining our model's accuracy. Each step in the training process is directly influenced by the quality of the preceding stage. That is to say, our model, as with other models, makes some assumptions about the input received at each step. Carefully choosing the right thresholds for the pre-processing stage such that unwanted noise is filtered out while keeping critical data largely untouched was one of the main initial roadblocks. Processed versions of the diaries page-to-page had a wide range of differences in the features of characters that were kept intact. While there were only slight, minute differences between some pages, there were glaring, extreme differences between others. Even the features of the same letters within the same page would vary greatly. Due to this problem, this may have had a large effect on our error calculations.

In its present state, the model that we developed showed a CER of approximately 18% when tested without post-processing. When analyzing the model's performance on a line of text extracted from one of Jewett's field diaries, one sample we selected was: "see The canyons of Desolation and" as the reference line to compare to the model's predictions.

Figure 5: "see The canyons of Desolation and"

Upon evaluating the model's prediction of "sese The canyois of Dratratin and," we noted an edit distance of seven from using the same Levenshtein distance algorithm discussed in the post-processing segment of this paper. It is important to understand that this edit distance is case sensitive. We translate this into a more common measurement of accuracy displayed in other papers, CER, by dividing our edit distance by the total number of characters in the reference string and turning it into a percentage. Further testing on a couple other lines extracted from the diaries reveals similar statistics of around 18% CER and around 39% Word Error Rate (WER).

Including the post-processing state, that same extracted line went from its original prediction to a prediction with only one character edit away from being completely accurate, "see The canyons of Desolatron and", which is a 3% CER. This was a massive success for part of our model, but this isn't the whole picture. Recall the original purpose and intent for this model's usage. Our model is able to perform well on common words and with the help of an extended dictionary containing species names that have appeared previously, it is able to do fairly well overall, albeit with some over-corrections. Our model's currently implemented post-processing correction method weights the first predicted character more than the others in order to achieve a better accuracy, but prior to that weighting it would change an incorrectly predicted word to another that shared very little visual similarity yet had one of the smallest

Figure 6: in the open glades Saw

Figure 7: Populus trichocarpa

distances from the target word. Our resulting average edit distance including post-processing was 11.6 characters for a page of writing.

5 DISCUSSION

Our goal was to train a machine learning classifier to transcribe Stanley Jewett's diaries. The main tools we used were pre-processing, Tesseract, and post processing. Prior to post-processing, our CER was 18.85%, and our WER was 39.86%, when trained for 3,000 epochs. We experimented with training for longer, however it resulted in a model that was over-fitting to our already small data-set. After post-processing, we had an average WER decrease of 3%. As compared to Drobac, S. and Lindén, K [7], who achieved a character error rate 1.7- 2.7%. The substantive differences in accuracy can be attributed to multiple factors. The most restrictive factor to the accuracy of our classifier is the size of the data-set. Ending with around three hundred and fifty training examples was an accomplishment, due to the fact the data-set was handmade. However, we believe strongly that given a much larger data-set, our classifier would have much better results. Moreover, another difference is the quality of their training samples. Although their samples have variation, they are still printed text. This allows for far less variation between samples of text. Our next step was to compare the performance of our classifier to an alternative method of archive digitization: human transcriptions.

In order to paint a complete picture of our concept for this project, we wanted to compare the results of our classifier to human attempts to transcribe these journals. This was in order to show the value of a model like ours to those who might question the efficacy of a model with a low accuracy score. Specifically, we wanted to see how students with a background in biology would perform when trying to transcribe these journals. In order to do this, we created a survey that was sent out to around thirty students. They were shown line images from a single page from one of Jewett's journals. This page contained common English words, as well as scientific Latin names. Figure 6 shows one example from this survey. This line transcribes to 'In the open glades saw'. Out of thirty four students, only one was able to transcribe this line exactly. The average character edit distance from this line was 5.56 characters, while our classifier had an edit distance of 3. To highlight the difficulty of transcribing these pages, figure 7 shows an example from the survey of a scientific name. This translates to 'Populus trichocarpa'. On average, the student's edit distance was eight characters. In contrast, after post-processing, we were able to transcribe this line

perfectly. After discussing our model's results and defending our proof of concept, we will now address the limiting factors of our project.

While our results are promising, there are multiple limiting factors to our model. The factors that could be affecting our performance the most is the size of our data-set, having relied on Tesseract's pre-trained model, and the novel content of the journals. A cardinal rule of training machine learning models is: the more data the better. Due to the fact we had a small data-set in a domain where there are many features for the model to consider, there was only so much we could do to stop the model from over-fitting. Without a larger data-set, it is far less likely that our model can adapt to examples of text different to what it has already seen. Due to the inconsistent nature of cursive handwriting, this likely is causing a substantive constraint to the performance of our model. The next limitation we would like to address is our reliance on Tesseract's engine. In many ways, the largest constraint we faced during this project was time. Ideally, we would have built a custom neural network, allowing us to have more control over convolutional layers and feature extraction. However, in a paper by Bagwe, S. et al. [8] that compared the accuracy of cursive transcription to handwritten print, after building a custom neural network, the cursive accuracy reached 65.7%. Therefore, it is not obvious if this would have provided better results. Finally, the last constraint of this project is the novelty of the content of the papers. While many of the pages contained common English words, a large portion of the diaries contained uncommon and outdated scientific words. These words were likely not built into the initial training of Tesseract, which likely caused a hit to our output.

There is still much to be done in order to transcribe Stanley Jewett's diaries that contain decades of data about biological systems around the Pacific Northwest. It is our hope that this project will continue in the future. With that, we will discuss some recommendations for anyone who may attempt to digitize his work in the future.

6 FUTURE WORK

We strongly believe that this project should be taken up by another group, and we are dedicated to leaving it in a state that could be easily picked up. To those that are interested in continuing our work here is a rough guide to follow. The first and most important step is to increase the size of the data set as that was the biggest limiting factor for our model's accuracy. The true efficacy of this project cannot be understood with a data-set of 350 training examples. In order to do this, more pre-processing needs to be done. We were able to pre-process the cleaner pages, the ones written in pen, to a sufficient degree that we could use them for training. However the vast majority of the pages are much more complicated. Challenges that those who would take on this project would encounter include: cramped and overlapping text, smudges, pages with drawings or maps, non-traditionally aligned text, sideways or upside down text, words that are truly illegible, and the "H" watermark. We have learned that any pre-processing is not a one size fits all, even ones that seem to be the same level of cleanliness. From there transcriptions of all the planned training pages should be made.

If choosing to stick with Tesseract as the OCR engine, before continuing on, it is important to make sure you are able to run Tesseract, as we ran into many difficulties just with file system placement and improper makefile commands built in. Once Tesseract is working, the next step is to create the line image, box file, ground truth trio for as many lines as deemed necessary. We used jTessBoxEditor to edit the box files, and while this program is sufficient, it is a time consuming process. Once all the data is run and the new model created, depending on the quality of the output, a form of post-processing should clean up the results.

We believe that the notes of Stanley Jewett should be open to the public, and a page on the University of Puget Sound's Museum of Natural History website should be made to store his writings. Furthermore, Jewett kept records of the birds that he saw and where he saw them, and many would benefit from these findings being added to the Ebird index. We hope that a new group can learn from our successes and failures to complete this project. We also want to recognize that Tesseract is not the only, or possibly, even the best way to create the model of Stanel G. Jewett's handwriting and would encourage any other groups to attempt other methods. We know it wont be easy to finish, but are sure that it will be rewarding.

7 CONCLUSION

We believe that Stanley G. Jewett's writings should be preserved due to their beneficial nature to the biologists and ecologists of the Pacific Northwest. We were given access to thousands of pages of field notes and journals written by him while he was in the field. Our project was to take these writings and use them to train an OCR engine capable of transcribing his inconsistent cursive. We began by deciding our project structure, where the main steps were pre-processing scans of text, and then using these to build a data-set that is compatible with our chosen engine, Tesseract 5. After using the data-set to train the model, we gave our model input. However, at this point the output was not near the accuracy we desired, so we added a final step to our pipeline: post-processing. Post-processing consisted using edit distance to be a glorified spell checker for our output. In the end, we were able to pre-process three journals consisting of 147 pages. By the time our work on the model came to a close, it was able to read some of Jewett's idiosyncratic style of writing. With just the beginnings of a data-set, when the model had predicted lines on a test page, there were approximately 11.6 edits on average away from the target answers. However, we ran out of time to officially transcribe a journal. We hope that this project is continued in the future, as we believe these journals contain valuable information that should not remain stale in our university's archive. With the groundwork we have laid out, we feel that if continued, there is much that could be achieved.

8 ACKNOWLEDGMENTS

We would like to extend our gratitude to professors Peter Wimberger, Adam A. Smith, and America Chambers for their unwavering support and guidance throughout the process of our research project. Professor Wimberger, we appreciate the enthusiasm and encouragement you brought to all of our weekly meetings. Your professional transcriptions and advice were invaluable to our project. Professor Chambers, thank you for generously giving us your time

and counsel that kept us motivated throughout the semester. Lastly, to Professor Smith, we are grateful for the consistent check-ins that helped us stay on track and meet our deadlines.

Our final acknowledgments go to VietOCR for creating jTessBoxEditor, which our project would not exist without, to Google for developing Tesseract and providing documentation, which was the very heart of our implementation, and to all the people that contributed to our project in our human study.

9 REFERENCES

[1] Ira N. Gabrielson. 1995. In Memoriam: Stanley Gordon Jewett. The Auk 73, 4 (Oct. 1956), 513–516. DOI, https://doi.org/10.2307/4081949

[2] A. El-Yacoubi, M. Gilloux, R. Sabourin, C. Y. Suen, An hmm-based approach for off-line unconstrained handwritten word modeling and recognition, IEEE PAMI 21 (8) (1999) 752–760.

[3] A. Poznanski and L. Wolf, "CNN-N-Gram for handwriting word recognition," in CVPR, 2016.

[4] B. Stuner, C. Chatelain, and T. Paquet, "Handwriting recognition using cohort of LSTM and lexicon verification with extremely large lexicon," CoRR, vol. abs/1612.07528, 2016.

[5] J. Sueiras, V. Ruiz, A. Sanchez, and J. F. Velez, "Offline continuous handwriting recognition using sequence to sequence neural networks," Neurocomputing, 2018.

[6] K. Dutta, P. Krishnan, M. Mathew and C. V. Jawahar, "Improving CNN-RNN Hybrid Networks for Handwriting Recognition," 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), Niagara Falls, NY, USA, 2018, pp. 80-85, doi: 10.1109/ICFHR-2018.2018.00023.

[7] Drobac, S. and Lindén, K. 2020. Optical character recognition with neural networks and post-correction with finite state methods. International Journal on Document Analysis and Recognition (IJDAR). 23, 4 (2020), 279–295.

[8] Bagwe, S. et al. 2020. Optical character recognition using deep learning techniques for printed and handwritten documents. SSRN Electronic Journal. (Jul. 2020).