Braille Translation Using a Keras-Based Language Model

C. Carr and M. Germanyan

Abstract

Braille image translation is a sector of natural language processing with great potential for aiding the community of those with visual impairments. Despite this, it remains a largely unexplored area of language modeling. In this project, we develop a Keras-based language model, building on optical character recognition and the noisy channel model to create an image translation system for images of Braille words. A neural net alone to translate Braille words at the character level has an extremely high error rate. With the addition of a language model, the accuracy of the translation increases dramatically. Though this remains an area that deserves much exploration, our models provide a strong introduction to the field of Braille translation through natural language processing, and show great promise for future development in the field. This paper explores both our Keras model as well as the addition of the language model, the improvement resulting from this combination, and future modifications that we speculate could improve accuracy even further.

Keywords: Braille image translation, natural language processing, Keras-based language model, optical character recognition, noisy channel model, neural net

Introduction

Braille is a touch-based reading system for people with visual impairments. It consists of raised dots on embossed paper that is read by moving the hands over the dots. There are two primary subsets of English Braille: contracted and uncontracted. In uncontracted Braille, each character represents one English letter or symbol. In contracted Braille, however, there are certain abbreviations for common words or letter combinations. For example, while the word "day" would appear as it does in Figure 1.1 for uncontracted Braille, in contracted Braille, it appears as in Figure 1.2. For the purposes of simplicity, this project focuses only on uncontracted Braille words of length 3.

Figure 1.1 Uncontracted Braille: 'day'

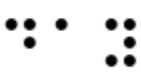


Figure 1.2 Contracted Braille: 'day'

This project sought to create an image translation system for English Braille. Our first model takes in images of English Braille words, segments them into their component characters, and translates them into the English Latin alphabet in a method reminiscent of optical character

recognition. Because the error rate on this character-to-character translation (and subsequently, reconstruction of those character translations into their original words) using this initial model was very high, the combination of these characters into words was unsuccessful. To lower these error rates, we also implemented a language model on top of this baseline model to reduce such errors.

Despite its widespread use within the blind community, Braille can be a highly inefficient method of communication. Due to its touch-based nature, there is little to no opportunity for summarization or "skimming" text written in Braille as there is for text written in a vision-based alphabet. An image translation model like the one we propose in this paper could serve to increase the ease and efficiency with which people read large amounts of Braille text, such as by providing a corpus of text to pass through another language model that can provide a text summary, information extraction, or text-to-speech technology.

Background

To our knowledge, no natural language processing model exists so far to perform a translation between English Braille images to the Latin alphabet. However, there are various natural language processing principles that we base our project on. For example, our multilayered model of image translation followed by a language model was partially inspired by the work of Kolak and Resnik, in which they propose using a noisy channel model to correct errors in optical character recognition (2002).

A noisy channel model is often used for correcting word misspellings. It treats the misspelled word as a distorted form of the true word, and finds the most likely correct word by comparing probabilities of all the words in a language model given the prior words in the sentence (Jurafsky & Martin 2020). We apply this noisy channel model to our project by implementing it at the character level. For our language model, we take the highest probability combination of characters given the previous characters (or combinations thereof) in the word.

Optical character recognition (OCR) enables the conversion of images of characters into editable data. It involves several steps, including scanning, segmentation, preprocessing, feature extraction, and recognition (Mithe, Indalkar, & Divekar 2013). In our project, we performed an OCR analysis on our Braille images by segmenting the word images into their component characters, preprocessing those character images so they were the correct size for our model, and training a Keras neural network to perform the feature extraction and recognition of those characters.

Methods

Keras Modeling

The first model we used was a Keras neural network. This was trained on data consisting of various images of Braille characters from Kaggle (Kumar 2020). These images include various copies of all 26 alphabet characters, as well as variations in orientation and brightness of the images. An example of the original character as well as each of these variations is shown in the Appendix Figures 2.1, 2.2, and 2.3, respectively. In validating the model on a subset of this data, we achieved 80% accuracy.

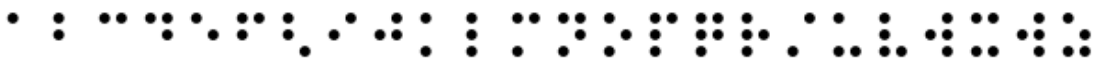
Figure 2.1 width height shift

Figure 2.2 Rotational shift

Figure 2.3 Brightness shift

Because we hoped our model would be generalizable to a wider variety of Braille images, we created our own testing dataset using images from a Braille translator website. An example of these images is shown in Figure 2.4. These images had different backgrounds than the images used to train/validate the Keras model, so the accuracy of the character-to-character translation of the model dropped substantially, to approximately 50%. This was after making some modifications to the original Keras model, including removing the character image variations such as rotation and brightness levels from the original Kaggle dataset, so they would more closely resemble the images we gathered from the Braille translator website.

Figure 2.4: Braille Translator of the alphabet



Despite this low accuracy, we attempted to translate a 3-letter Braille word using this Keras model. The program reads in an image of a word (which we created using the Braille website from 3-letter words in the Brown corpus, from the nltk library in Python), segments it into its component characters based on the equally-spaced whitespace between the individual characters, then feeds each character image individually into the Keras model. Once the Keras model outputs its predicted character based on the highest probability, the program reassembles those predictions and outputs the resulting word, without considering the surrounding characters. For example, for an input word “hyp” in Braille, the Keras model predicted the output “hjp” because individually, the characters “h”, “j”, and “p” were assigned the highest probability.

Language Modeling

Because this accuracy was so low, we decided to implement a language model in addition to the original Keras model. Our model focused on 3-letter words, so we created dictionaries where the 1, 2, 3, and 4-grams of characters were the keys (the fourth character being a stop character, "</w>", added manually to the end of each word used in the language model), and their counts being the values. We could then use these dictionaries of counts to calculate the probability of any Latin character given the characters that had preceded it. These dictionaries were created using all words in the Brown corpus from nltk that did not include punctuation or numbers. Take, for example, the language model prediction for the word "the". In the output of this word, the first output would correspond to $p_{lm}(t)$, the second output would be $p_{lm}(h|t)$, the third output would represent $p_{lm}(e|th)$, and the final output would equate to $p_{lm}(</w>|the)$.

Combining the Models

The next step was to combine the Keras model with the language model. For each character inputted to the Keras model, we pulled the 5 letters with the highest probability, resulting in 5 characters outputted with their associated probabilities for each of the 3 characters in the word. This resulted in a total of 125 character combinations.

We then passed each of these 125 letter combinations through our language model. The language model was able to take into account the context of the word in that it would calculate the probability of a character given the previous characters in the word, which we hypothesized would improve the accuracy of the model's predictions.

Once we had the output probabilities for the Keras model and the language model for each of these 125 words, we multiplied them together pairwise. See Equation 2.7 for an example of combining the probabilities for the Keras-outputted word "hjp". In this equation, p_k refers to the probability given to that character by the Keras model, and p_{lm} refers to the probability given to the character or combination of characters by the language model.

Equation 2.7: Calculating the Combined Probability of "hjp"

$$p_k(h) * p_{lm}(h) \times p_k(j) * p_{lm}(h|j) \times p_k(p) * p_{lm}(p|hj) \times p_{lm}(/w|hjp)$$

The final output predicted word was the word with the highest combined probability when calculated in this way.

Results and Discussion

We tested our Keras based language model on all three letter words from the nltk package words corpus of which consisted a total of 1293 words. With this version of the Keras model, our accuracy reached only 6.7% on our test images. We then used character images from our test set

to create the braille input images based on the words in our test set. The addition of the language model improved our accuracy to 17.1% (a nearly 300% increase compared to the accuracy of the Keras model alone). The model's improvement can certainly be attributed to the addition of the context-dependent language model; given the combined probability formula in Figure 2.7, a letter predicted by the Keras model can be virtually eliminated by the language model if the combination of letters is uncommon in the English language. For example, in the predicted word "hjp", the combination of letters "hjp" is essentially nonexistent in English words, especially at the end of a word. Therefore, $p_{lm}(</w>|hjp)$ will have an extremely low (possibly 0) probability. This in turn results in the combined probability for this prediction being extremely low. In this case, "hjp" would no longer have the highest probability, and therefore would no longer be the final predicted word.

Looking at Figure 3.1, we can see examples where the keras model alone wrongly predicts the word, but the addition of the language model helps predict the correct word. Figure 3.2 demonstrates the results of a small sample of size 10 on our model. Out of the 10 words, our model correctly predicted 4 of them. Furthermore, it is worth noting that those words that were wrongly predicted were mostly off by one character, which was also often the case with the Keras model alone, as is shown in Figure 3.1.

Figure 3.1: Word Predictions from Keras with and without Language Modeling

actual	keras_model	language_model
til	jil	til
nap	jap	nap
ife	ipz	ife
was	jas	was
ade	adz	ade

Figure 3.2: Predicted vs. Actual Words for Combined Modeling

```
Predicted ['dit', 'via', 'pon', 'pip', 'his', 'pip', 'apa', 'ped', 'xas', 'sex']
Actual ['dit', 'via', 'led', 'fip', 'nub', 'pop', 'ara', 'ped', 'mas', 'sex']
```

Conclusion

Braille image translation remains an area of natural language processing that warrants a great deal of attention. Our work has demonstrated that significant improvements can be made to Braille optical character recognition through the addition of a context-dependent language model, and we are extremely content with the results of this project. Making a more powerful language model, such as through the implementation of a recurrent neural network, specifically a long short term memory network, would likely make this model much more accurate, and

therefore, much more useful to the Braille-reading community. Other methods of improving the accuracy of this model include improving the accuracy of the Keras model prediction at the character level by modifying hyperparameters, as well as training on more data. In addition, although our work has limited our translation to only images of 3-letter words, one can easily see how this kind of technology could be expanded to words of any length, ideally to whole sentences, and even be expanded to work in translating contracted Braille words or sentences. At the sentence level, the language model could become even more context-dependent by taking into account the surrounding words of a target word, rather than only the characters of that target word. If the accuracy of this kind of modeling becomes sufficiently high, this would provide a pathway for the addition of speech-to-text technology, information extraction models, and text summarization, all of which would help with increasing the efficiency of communicating with Braille.

References

Braille Translator. (n.d.). Retrieved from https://www.brailletranslator.org/

CodeBreaker619. (2021). Braille Image Classifier using Neural Networks. Retrieved 02/18/2021 from https://www.kaggle.com/codebreaker619/braille-image-classifier-using-neural-networks

Kolak, O., & Resnik, P. (2002). OCR error correction using a noisy channel model. Proceedings of the Second International Conference on Human Language Technology Research -, 257–262. https://doi.org/10.3115/1289189.1289208

Mithe, R., Indalkar, S., & Divekar, N. (2013). Optical Character Recognition. Optical Character Recognition, 2(1), 4.

Shanks. (2020). Braille Character Dataset. Retrieved 02/18/2021 from https://www.kaggle.com/shanks0465/braille-character-dataset