OCR Spell Check Independent Study (SPIN NCSA)
David Zhu
Prof. Jill Naiman
Jul. 18th, 2022

Final Report of OCR Spell Check SPIN Research Study

Abstract

Optical Character Recognition (OCR) is one of the methods to extract text from an image. There are many issues with the accuracy of OCR such as spelling errors and punctuation[1].
My study aims to investigate the types of OCR errors produced with the Tesseract OCR engine and quantify the accuracy of the OCR engine on astrophysical historical documents through different optimization of the OCR process.

Introduction

What is OCR:



Figure 1 – an example of an OCR page, left is the actual scanned image, and right is the OCR result

OCR stands for optical character recognition, which is the technique to convert a pixelated image into words and characters for better readability, and interpretability. Some ancient books are only available in scanned versions so some words may be hard to see, as well as they are hard to preserve digitized. In this project, we explored different ways to examine how to increase

[1] A Prototype Gutenberg-HathiTrust Sentence-level Parallel Corpus for OCR Error Analysis: Pilot Investigations. Mjiang et. al.

the readability and correctness of OCR data so they can read better to humans. An example of an OCR'd page is in the figure above (figure 1).

Our goal is to digitize the old scanned paper and eventually host figures and their captions at Astronomy Image Explorer[2] for people better to search and index that information.

Literature Review

From *Practical natural language processing: A comprehensive guide to building real-world Nlp systems***,** We read about natural language processing (NLP), we interacted with different NLP techniques and models and learned how to practice machine learning in NLP in python notebooks. This book provided us a foundation for the future machine learning related works.

This study was based on the research of *Survey of Automatic Spelling Correction by Daniel Hládek et. al.* talks about the approach of different spelling correction models. In this article, Hládek talked about the underlying algorithm behind major OCR engines and their performance. By understanding these algorithms, we can present some analytics to discover ways of improving the OCR engine.

In *A Prototype Gutenberg-HathiTrust Sentence-level Parallel Corpus for OCR Error Analysis: Pilot Investigations* by mjiang et. al., they provided us with some metrics about WER and CER to quantify the performance of the OCR engine. These metrics became the major index of analyzing error rates in our study.

Methodology

We build our engine and test cases on google colab. For the OCR engine, we choose the google OCR engineTesseract, as implemented with the Python wrapper named "pytesseract"[3].

Our examination and exploration extended through different methods and modules such as using a spell checker[4], using GROBID data format to parse PDF, using metric measurements to create our own dictionary about the topic, and ways to quantify word error rate (WER, which means how many words are parsed error from ground truth) and character error rate (CER, which means how many characters are parsed wrong).

The word error rate (WER) and the character error rate (CER) are quantifiers to show how many errors occur in an OCR result. The formula is the sum of edit operations (substitution, deletion, insertion) divided by the sum of all words/characters. WER looks at word-level scope while CER looks at character scope.

---

[2] *Astronomy Image explorer*. Astronomy Image Explorer. (n.d.). Retrieved July 22, 2022, from http://www.astroexplorer.org/

[3] *Pytesseract*. PyPI. (n.d.). Retrieved July 22, 2022, from https://pypi.org/project/pytesseract/

[4] Filyp. (n.d.). *Filyp/autocorrect: Spelling Corrector in python*. GitHub. Retrieved July 22, 2022, from https://github.com/filyp/autocorrect

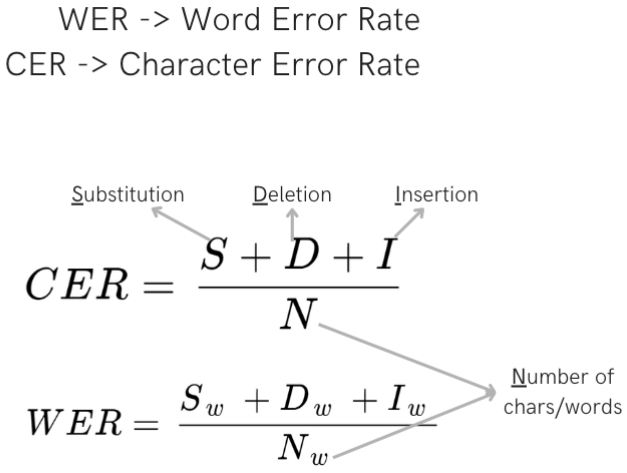


Figure 2 - the formula of WER and CER; Sw, Dw, Iw are substitutions, deletions, and insertions in the word level.

Results

First, we applied the Tesseract OCR process (using pytesserect[5]) on several scanned pages to get the word output and its confidence score. In figure 3, we highlighted in red some textboxes that pytesserct does not feel so confident so we can have an overview of what are common errors. There are also text boxes where OCR feels confident but an actual error, such as “Fic” and “Fig” in figure 3. These kinds of results suggest that math formulas and letters not-in-line are frequently OCR'd incorrectly.

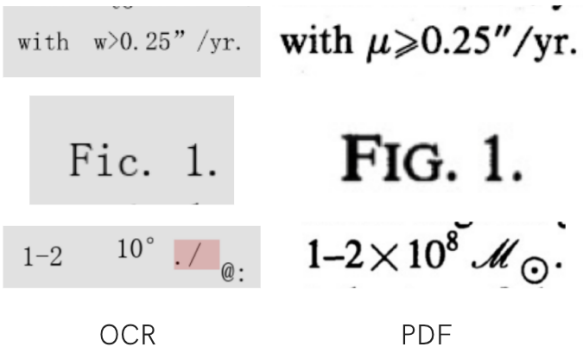


Figure 3 - OCR errors with their ground truth

[5] *Pytesseract*. PyPI. (n.d.). Retrieved July 22, 2022, from https://pypi.org/project/pytesseract/

Then we tried to look at each character to see which character has a higher chance of OCR'ed wrong. I used the Levenshtein Distance metric to sort each edit operation and count the occurrence of each wrong character divided by the total occurrence of it. The right side figure shows the most common pairs of errors among these papers.

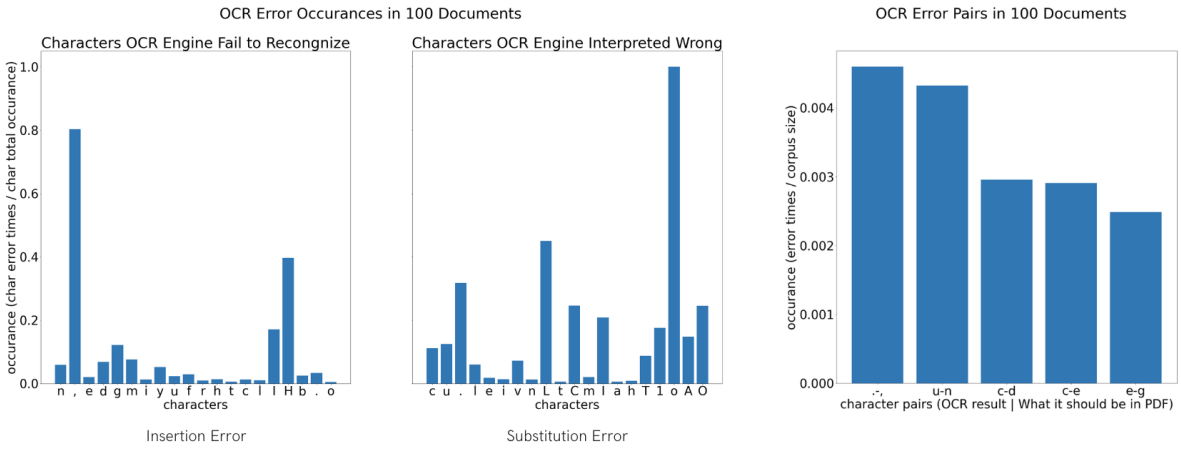


Figure 4 -
left: The character level error frequencies; the left side is characters OCR fails to recognize, and the right side are characters OCR engine interpreted wrong;
right: top 5 frequently OCR error pairs; the left part in a pair is the OCR result, and the right part in a pair is what it should be as ground truth

Conclusion

By investigating error rates and frequencies we can be clear about what are the essential letters for OCR. From Figure 4 for our default run of Tesseract, we see that "." is often confused with ",". In future work, we will quantify how often this occurs when we modify the parameters of Tesseract.

We have interacted with different types of correctors modules in Python for minimizing errors in OCRed text. These methods are helpful in making OCR data more correct, while they all have some ways to improve.

Future Works

The next step is to compare how different resolutions and file formats affect the performance of the OCR engine; and how different OCR configurations and language models would change the result (as in figure 5). Digitizing a larger corpus of documentation still needs work on implementing different deep learning models and methods for OCR correction.

Page segmentation modes:

0. Orientation and script detection (OSD) only.
1. Automatic page segmentation with OSD.
2. Automatic page segmentation, but no OSD, or OCR. (not implemented)
3. Fully automatic page segmentation, but no OSD. (Default)
4. Assume a single column of text of variable sizes.
5. Assume a single uniform block of vertically aligned text.
6. Assume a single uniform block of text.
7. Treat the image as a single text line.
8. Treat the image as a single word.
9. Treat the image as a single word in a circle.
10. Treat the image as a single character.
11. Sparse text. Find as much text as possible in no particular order.
12. Sparse text with OSD.
13. Raw line. Treat the image as a single text line, bypassing hacks that are Tesseract-specific.

OCR Engine modes:

0. Legacy engine only.
1. Neural nets LSTM engine only.
2. Legacy + LSTM engines.
3. Default, based on what is available.

Figure 5 - different tesseract configurations and language models. Entries in grey would be first considered common configurations

References

Astronomy Image explorer. Astronomy Image Explorer. (n.d.). Retrieved July 22, 2022, from http://www.astroexplorer.org/

Astrophysics data system. NASA/ADS. (n.d.). Retrieved July 22, 2022, from https://ui.adsabs.harvard.edu/

B., S. V., Majumder, B., Gupta, A., & Surana, H. (2020). *Practical natural language processing: A comprehensive guide to building real-world Nlp systems*. O'Reilly Media.

Filyp. (n.d.). *Filyp/autocorrect: Spelling Corrector in python*. GitHub. Retrieved July 22, 2022, from https://github.com/filyp/autocorrect

Hládek, D., Staš, J., & Pleva, M. (2020). Survey of automatic spelling correction. *Electronics*, *9*(10), 1670. https://doi.org/10.3390/electronics9101670

Jiang, M., Dubnicek, R. C., Worthey, G., Underwood, T., & Downie, J. S. (2022). A prototype Gutenberg-hathitrust sentence-level parallel corpus for OCR error analysis. *Proceedings of the 22nd ACM/IEEE Joint Conference on Digital Libraries*. https://doi.org/10.1145/3529372.3533298

Leung, K. (2021, September 23). *Evaluate OCR output quality with character error rate (CER) and word error rate (WER)*. Medium. Retrieved July 22, 2022, from

https://towardsdatascience.com/evaluating-ocr-output-quality-with-character-error-rate-cer-and-word-error-rate-wer-853175297510

Pytesseract. PyPI. (n.d.). Retrieved July 22, 2022, from https://pypi.org/project/pytesseract/

Rijhwani, S., Anastasopoulos, A., & Neubig, G. (2020). OCR post correction for endangered language texts. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. https://doi.org/10.18653/v1/2020.emnlp-main.478