Improving Tesseract OCRs performance via retraining and character segmentation

Ayush Sharma, Nischal Nadhamuni
Massachusetts Institute of Technology
Cambridge, MA
{ayushs, nischal}@mit.edu

Abstract

Optical Character Recognition is a challenging Computer Vision problem, and even the state of the art open-source solutions aren't completely robust to variances in scanned text. In this paper, we improve and build upon Tesseract, Google's open source OCR engine with the goal of improving it's performance on legal documents. The proposed strategies are data-augmentation coupled with retraining, as well as employ pre-processing methods to increase the prior probability of a better OCR output result. Specifically, we retrain Tesseract on a custom legal document dataset, and implement a heuristics based character segmentation method.

1. Introduction

Tesseract, is an OCR engine that's been open sourced and developed by Google since 2006. [4] Despite being a fairly reliable software, Tesseract however does suffer from multiple issues such as loss of accuracy due to noisy data, borders, faded text, bi-columnar text, broken as well as conjoined words. For our project we set the goal of improving Tesseract's performance on a specific type of document: legal contracts. By constraining the problem in this way we were able to eliminate variance in our dataset and implement solutions that led to more significant improvements to accuracy. We decided to focus our efforts on two specific distortions that led to significant reductions in Tesseract's accuracy - faded text in documents and conjoined characters in words.

Individual work - Ayush
Ayush worked on the following components of the project -

- Wrote custom fading module for image augmentation
- Devised and implemented Character Segmentation Algorithm
- Researched methods for evaluating results including edit distance and text comparison.

2. Related Work

OCR is an active field of research and much previous work has been done.

Work of E. Borovikov [2] surveys advancements in OCR techniques. However, even modern OCR engines aren't robust against text distortions. In this paper, we propose and implement improvements for two non-trivial distortions.

In addition, paper [6] surveys various existing methods of character segmentation. However, as we present in our paper, a simple heuristics based approach may get decent results. In addition, we also discuss the possibility of a supervised learning formulation for our method which would be a novel technique and would be interesting to evaluate.

Much literature also exists on techniques that may be applied to improve the accuracy of Tesseract in particular [7]. However we found that most of these are not of use to the specific use case of scanned legal documents that we were targeting.

3. Approach

3.1. Identifying existing shortcomings

The first objective we set ourselves was to establish the shortcomings of Tesseract on contracts. In order to do this we first had to gather a representative set of sample data and establish it's baseline accuracy. We found and used a set of 44 publicly available scanned contracts. These contracts were not all drawn from the same source and had significant variance in terms of scan quality, content, structuring, and other factors.

We then set out to measure the baseline accuracy of Tesseract on these documents. Measuring the accuracy

of OCR systems is far from a trivial problem. We first considered a series of approaches that would require manual annotation of this data. We initially decided to use Tesseract's HOCR output format for measuring accuracy. This is a format where each word is encapsulated in a '¡p¿' tag and it's exact coordinates within the page are embedded in the tag. However, upon further investigation we found a critical flaw in this approach. Tesseract often incorrectly split words, making it very difficult to precisely compare each word in Tesseract's output to a corresponding word in the annotated data. This undermined the whole purpose behind using HOCR in the first place.
We then decided to try a method that would be more resilient to 'off-by-one' type errors; using the edit distance between Tesseract's output and the annotated document. This method performed to expectations. However, we soon realized how slow (and potentially expensive) manually annotating (transcribing) scanned documents would be. Thus we decided to investigate methods that would not require manually annotated data
A number of past research papers around OCR engines have used classifier confidence as a measure of accuracy [1]. Tesseract is also able to produce confidence metrics. Accordingly, we produced confidence metrics for all 44 scanned documents. However, as we compared the classifier text output with the confidence score it was abundantly clear that confidence and accuracy were not tightly correlated on our dataset.
Having already spent a lot of time trying out different methods of finding a baseline, we decided to simply do a visual inspection of Tesseract's output on each of the 44 documents. Rather than consistent performance across documents, we saw clear drops in text output quality for some documents and for some parts of documents. Based on this analysis we identified a number of confounding document features that we're producing difficulty for Tesseract. The most significant of these were:
Faded text, Conjoined words, Notes within text, Signatures, Redlining. Mixtures of landscape and portrait oriented pages

We decided to work on solving the problems of faded text and conjoined words. These were problems that appeared fairly frequently and had significant negative impacts on Tesseract's performance. Additionally, these problems seemed more interested to us, from a computer vision standpoint, than many of the other issues.

3.2. Conjoined Characters

The problem of conjoined words occurs when poor scan quality results in certain characters being joined together. Our manual review revealed that this was a significant source of error for Tesseract. Figure 1 illustrates the scope of the problem.

(a) caption (b) caption

Figure 1: Conjoined Words Example

For evaluating our performance, our approach was to segment a document into multiple images, each containing a single word. We would then gather annotations for each of these images and use this to measure accuracy.
In order to accomplish this we first extracted sections of documents with high densities of conjoined characters. We then outsource their segmentation into individual images and annotation on UpwWork.com. In this way, we were able to gather 1200 annotated images (for $20).
The specific algorithm used for character segmentation and the results achieved are discussed in further depth in section 4.2

3.3. Faded text

The problem of faded text arises from poor scan quality. Our baseline analysis showed that it was one of the largest factors in determining the quality of output produced by OCR. As can be seen below, the text fading can render characters nearly indistinguishable from one another

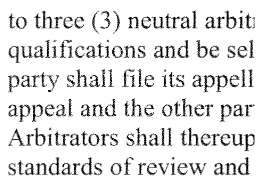

to three (3) neutral arbit
qualifications and be sel
party shall file its appell
appeal and the other par
Arbitrators shall thereup
standards of review and

Figure 2: Example of faded words

Our approach to tackling this problem was to retrain Tesseract using the Training Tesseract module [5]. This module was originally intended for training Tesseract on new languages and new fonts, but we wanted to apply it to the problem of fading to observe possible outcomes.
In order to train Tesseract, one must provide a series of images from which Tesseract will extract a boxfile. A boxfile is a file in which each line displays a single character along with it's pixel position in the document. This file format forms the annotated data for the classifier.
Typically users are advised to train on small amounts of data (on the order of a few hundred characters) and manually edit the boxfile generated by Tesseract. However, we wanted to train on much larger amounts of data so this approach would be prohibitively slow.
From previous research done by Nischal we had access to

the SEC EDGAR database of 900,000 legal contracts in both docx and pdf formats. It must be noted that the pdfs in this dataset are pristine as they were directly generated from the docx files. We used this dataset to train Tesseract in the following way:
We generated boxfiles with Tesseract using the pdfs mentioned above. In order to verify that Tesseract was actually performing near perfectly, we measured the Levenshtein edit distance between Tesseract's output on the pdf and the text, as directly extracted from the original docx. This test confirmed that Tesseracts accuracy (and therefore the accuracy of the generated boxfiles) was near perfect (upwards of 99.5% on all documents.).
We then augmented the input pdf using an image fading technique that will be discussed shortly, and passed the augmented pdf into the training module along with the boxfile. In this way we were able to generate a perfect boxfile from a faded image in very little time.
Our choice of fading algorithm was very important for this as it would have to closely mirror the type of fading found in actual scanned documents. We first tried uniformly fading the images, by increasing the pixel values across-the-board. However (as shown below), this produced images that just looked 'lightened', rather than faded. Tests with Tesseract on this revealed that such fading actually posed no difficulty to the classifier and was therefore obviously not an accurate representation of truly faded scanned documents.
Our next approach iterated over all pixels and changed the pixel value to 255 (white) with some pre-defined probability. This created images that much more closely resembled the original faded text, as can be seen below.
We then proceeded to train a classifier and test it as will be described in section 4.1.

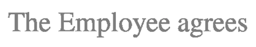

The Employee agrees

Figure 3: Words faded with uniform whitening

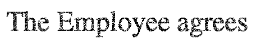

The Employee agrees

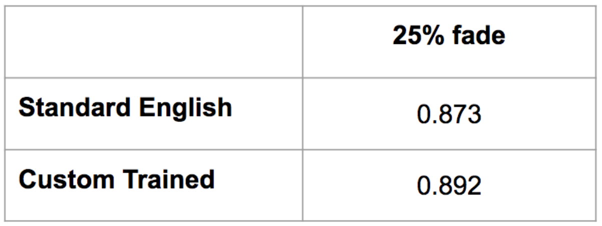
Figure 4: Words faded with our custom script

4. Experiments and Results

Based on the issues we identified and decided to work on, we experimented with the following methods details of which are presented in this section.

4.1. Performance of retrained Tesseract OCR Engine

In order to test our newly trained classifier, we needed annotated (transcribed) faded documents. Owing to the difficulty of manually transcribing documents, we resorted to using the same fading algorithm described in section 3.3. We acknowledge that this may introduce bias into our system depending on how well our fading algorithm mimics the documents that were actually faded at time of scanning. Adopting our approach made transcription a trivial task as we already had the docx versions of the data available to us and just had to extract the text from them.
We decided to use Levenshtein edit distance as our accuracy metric. We calculated percentage accuracy as $(1 - lev_dist/len(doc)) * 100$
When a 25% fade was applied to the test set, our newly trained model performed as shown in Figure 5. As we increased the amount of fading on the test set we observed marginally better performance of our classifier compared to Tesseract's standard English language classifier. However, at 50% fading, there was an extremely sharp drop in the accuracies of both. Upon inspection, both models were creating completely garbled output. Unfortunately, due to time constraints, we were unable to properly investigate this.



	25% fade
Standard English	0.873
Custom Trained	0.892

Figure 5: Tesseract English model vs custom model on faded text

4.2. Resolving conjoined words via Character Segmentation

Our approach to this problem was loosely based on dissection method [6] Our algorithm is based on heuristics and works as follows –

```
for columns in image:
    is_word_boundary(threshold,
    margin_width)
    insert_whitespace(width)
```

Basically, we iterate over the columns, detect whether the current columnar region is a word_boundary, and if so, insert some small number of whitespace columns. The tricky part here is how to detect whether a columnar region in the array from the image is belongs to a word boundary. By the construction of our problem, our use case was constrained to solving the character segmentation given individually segmented words. Therefore, we relied on a heuristics based approach for detecting character boundary based on the ratio of text/foreground pixels. If the ratio is lower than the threshold value, we categorize it as a word boundary and separate the word boundary further.

Character segmentation is a challenging problem since there's two competing distortions that we need to account for:
1. there's the conjoined words, that is, words that are mushed together, but we also have
2. individual characters that are split. The text corpus contains both kinds of distortions, and therefore the algorithm described above, if applied naively, performs worse. Lowering the threshold results in more conjoined characters getting separated, however it also starts splitting individual characters into further meaningless chunks of pixels.

To account for these competing effects, we implemented a grid search method to tune the hyper-parameters of this model, namely the threshold and margin_width. Specifically, we search over a range of values for both parameters, evaluate the model's performance as measured via the levenshtein distance metric, and choose the one with lower average error. While tuning the hyper-parameters, one would want to keep in mind that the metric for model evaluation may be different for different use cases and varying distributions of datasets.

Results: Figure 6 below shows the progression of cumulative levenshtein distances across both the distorted (conjoined) text and character segmented text in our word corpus. The average accuracy as summed over the entire corpus increases after applying the preprocessing routine of character segmentation. In addition, we see that some words are recognized in a way better fashion after going through the character segmentation stage.

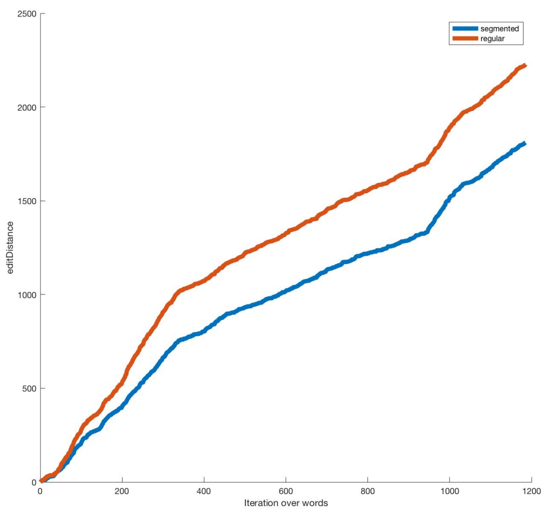


Figure 6: Segmented words vs Conjoined Words

forth
fo r t h

(a) joined (b) segmented

Figure 7: Conjoined vs Segmented Word

Notice that splitting the words into characters in this way makes the characters more identifiable - this property is reflected in Tesseract's OCR stage as well where these words are recognized closer to the ground truth (measured against human labelled dataset).

5. Discussion

Competing distorting effects in character segmentation –

While our character segmentation method performs fairly well on certain types of conjoined character distortion, it does suffer from a limitation of chopping up characters in some other cases - this happens when the original word has characters that aren't continuous in which case the margins are accidentally picked up as actual character margins. Figure 8 shows an example of this issue - -

(a) joined (b) segmented

Figure 8: Character splitting

Next, we describe a line of attack for future improvement on this -

Character segmentation as a supervised learning problem —
As described in section [INSERT REF], our custom character segmentation method containts hyper-parameters which we are currently optimizing via a simple grid search. However, we also came up with a machine learning formulation of this problem. Essentially, the idea is to learn the parameters instead of manually tuning them. We calculate the loss based on distance from actual transcribed word.

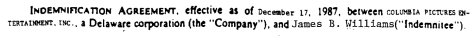
$$L_{model} = Loss[Z(\theta_{w,m,t}) - label]$$

Labelled samples serve as training examples, and the model parameters are finally optimized using a stochastic gradient descent minimizing the validation loss. Note that this would be different approach than the existing transfer based learning method [3]

Although we didn't have time to fully test this approach, but we certainly believe this would be a good direction for future work.

Partially faded documents –

While our approach to tackling fading works well at a document-wide level, it is often the case (as seen below in figure 8) that only certain sections of a document suffer from fading. In this case one would require a mechanism to intelligent detect faded areas and selectively classify those areas using the custom trained model described in Section 3.3, or something similar. In general, mechanisms to mitigate the effects of certain document features must also be paired with a methodology to detect the presence of those features in the first place. Due to time limitations, we were unable to construct such a mechanism but we do believe it is a promising avenue for future work.



INDEMNIFICATION AGREEMENT, effective as of December 17, 1987, between COLUMBIA PICTURES ENTERTAINMENT, INC., a Delaware corporation (the "Company"), and James B. Williams("Indemnitee").

Figure 9: Partially faded sections of text

6. Conclusion

Google's Tesseract platform is a powerful solution, particularly given its flexibility and adaptability as evidenced by modules such as Tesseract-Train. However, when constrained to a single domain (in this case legal documents), a number of challenges arise which Tesseract is not necessarily resilient against. When considered individually, many of these problematic document features can be partially or fully resolved (as this paper has shown in the cases of faded text and conjoined characters). However in order for such solutions to be applicable in a production environment, they must be paired with some mechanism that is able to dynamically detect and apply these solutions to only those parts of a document where they are necessary. This reality also gives some insight into why such modifications are not built in to OCR systems; they are not universally applicable across domains and would almost certainly cause reductions in accuracy is applied everywhere.

References

[1] S. Bapat. Determining confidence in optical character recognition resolution using bayesian networks.

[2] E. Borovikov. A survey of modern optical character recognition techniques. `https://arxiv.org/abs/1412.4183`, 2014.

[3] N. F. E. Kavallieratou, E. Stamatatos and G. Kokkinakis. Character segmentation using transformation based learning.

[4] Google. Tesseract ocr engine. https://github.com/tesseract-ocr/tesseract, 2015.

[5] o. s. Google. Retraining tesseract using language data. `https://github.com/tesseract-ocr/tesseract/wiki/Training-Tesseract`.

[6] E. L. R.G. Casey. A survey of methods and strategies in character segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume: 18, Issue: 7, Jul 1996), 1996.

[7] zdenop. Improving tesseract. https://github.com/tesseract-ocr/tesseract/wiki/ImproveQuality, 2015.